

BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

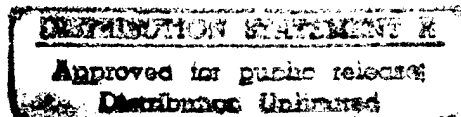
THESIS

AUTOMATIC IDENTIFICATION OF REMOTE ENVIRONMENTS
AND CALIBRATION OF VIRTUAL MODELS

by

TIMOTHY M. SCHULTEIS

Bachelor of Science in Engineering
Mechanical Engineering
University of Pennsylvania
Philadelphia
1992



Submitted in partial fulfillment of the
requirements for the degree of

Master of Science

1997

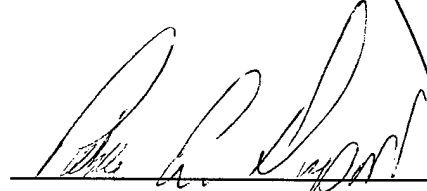
DTIC QUALITY INSPECTED 1

19970210 042

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 5 Feb 97		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE Automatic Identification of Remote Environments and Calibration of Virtual Models			5. FUNDING NUMBERS	
6. AUTHOR(S) Timothy M. Schulteis				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Boston University			8. PERFORMING ORGANIZATION REPORT NUMBER 97-006	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DEPARTMENT OF THE AIR FORCE AFIT/CIA, BLDG 125 2950 P STREET WPAFB OH 45433-7765			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)				
14. SUBJECT TERMS			15. NUMBER OF PAGES 79	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT	

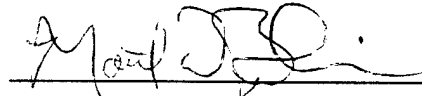
Approved by

Advisor:



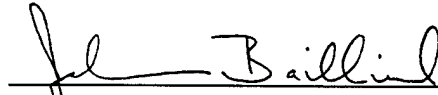
Pierre E. Dupont, Ph.D.
Associate Professor of Aerospace and Mechanical Engineering

Second Reader:



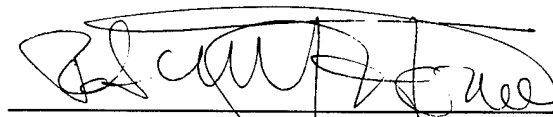
Matthew D. Berkemeier, Ph.D.
Assistant Professor of Aerospace and Mechanical Engineering

Third Reader:



John B. Baillieul, Ph.D.
Professor of Aerospace and Mechanical Engineering
Professor of Manufacturing Engineering

Fourth Reader:



Robert D. Howe, Ph.D.
Associate Professor of Engineering and Applied Sciences
Harvard University

Acknowledgments

The author would like to thank ...

Prof. Pierre Dupont for all his time, guidance, help, and encouragement in completing this research. Without him, this could never have been accomplished.

Prof. Rob Howe for his assistance in this research and the use of the teleoperated hand system and other laboratory equipment.

Paul Millman for his work in helping to collect and analyze the data from many of the experiments.

Parris Wellman for his explanations of the details and inner workings of the teleoperated hand system and providing the basis code used for creating the virtual environment.

Professors Baillieul, Berkemeier, Dupont, and Howe for serving on the thesis committee.

The **United States Air Force** and **Boston University** for providing this opportunity to obtain a graduate degree.

...and finally the whole gang of fellow graduate students, especially **Karen, Adam, Gretchen, Prakash, Josh, Trevor, Brian, Becca, Kate, and Dave** for collaborating and commiserating on all that went well and all that did not.

AUTOMATIC IDENTIFICATION OF REMOTE ENVIRONMENTS AND CALIBRATION OF VIRTUAL MODELS

TIMOTHY M. SCHULTEIS

Boston University, College of Engineering, 1997

Major Professor: Pierre E. Dupont, Associate Professor of Mechanical Engineering

ABSTRACT

A method is presented to automatically estimate parameters and identify constraints of remote environments in a teleoperated robot system and then to use that information to create virtual models of the remote environment. Such a model can be used in virtual training systems and in teleoperated systems with time delays. In addition, automatic parameter estimation can help the operator to improve task performance time and safety, and to plan object handling strategies. Ideally the system would determine these parameters while normal tasks are performed without interference in task accomplishment. The parameters are of two types: inherent object properties, which include weight, size, coefficient of friction, etc., and object/environment constraint parameters (i.e., parameters describing the trajectory of constrained motion and forces applied by the constraints.) In addition to constraint parameters, constraint identification includes determination of whether the constraint exists and development of the constraint model.

Contents

List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Application Examples	3
1.2 Background	5
1.2.1 Three Sub-Problems of the Environment Identification Problem . . .	6
1.3 New Work and Thesis Outline	8
2 Literature Review	10
2.1 Task Decomposition	10
2.2 Segmentation	11
2.2.1 Qualitative Reasoning with Thresholding	12
2.2.2 Hidden Markov Model	13
2.3 Parameter Estimation	15
2.4 Calibration of Haptic Feedback Systems	16

3	Theory of Parameter Identification	17
3.1	Task Decomposition	17
3.2	Segmentation	19
3.2.1	Stage 1	20
3.2.2	Stage 2	21
3.2.3	Stage 3	22
3.3	Parameter Estimation	23
3.3.1	Estimation Difficulties	24
4	Constraint Identification and Modeling	25
4.1	What is a Constraint?	26
4.2	Identifying the Presence of a Constraint	28
4.3	Modeling of Constraints	29
4.3.1	Spline Representation	29
4.3.2	Joint / Screw Representation	30
4.3.3	Center of Rotation Representation	32
4.4	Issues in Determining the Best Representation	32
5	Experimental Results	35
5.1	Teleoperated Dextrous Hand System	35
5.2	Parameter Identification: Block Stacking Task	37
5.2.1	Task Decomposition	38
5.2.2	Segmentation	38
5.2.3	Parameter Estimation	44

5.2.4	Results of Parameter Identification for Block Stacking Task	47
5.3	Constraint Identification	50
5.3.1	Crank Turning: Identification of a Rigid Constraint	50
5.3.2	Needle Insertion: Identification of a Configuration Dependent Non-rigid Constraint	53
5.4	Development and Calibration of Virtual Model	56
5.4.1	Modification of Two-fingered Hand System	56
5.4.2	Block Calibration	57
5.4.3	Crank Calibration	62
6	Conclusions	64
6.1	Summary of Thesis	64
6.2	Future Work	65
A	Control Code for Virtual Simulation	68
A.1	VROBOT.C	70
A.2	BLOCK2.C	73
	Bibliography	77

List of Tables

3.1	Qualitative Values of Sensed Parameters Which Define Task States	22
5.1	Relevant Signals and System States for Estimation of Object Parameters	
	During the First Four Subtasks	39
5.2	Qualitative Values of Sensed Parameters which Define Task States	40
5.3	Methods of Property Estimation	46
5.4	Comparison of Estimated (Mean and Standard Deviation) and Actual Envi-	
	ronment Parameters	47

List of Figures

1.1	Flow Diagram of the Parameter Identification Problem.	2
1.2	Block Diagram of a Teleoperation System with Property Estimator.	6
1.3	The Three Parts of the Environment Identification Process Applied to a Pick-and-Place Task.	7
2.1	Example of the Thresholding Technique. When the raw data exceeds a given threshold value, the value of the resulting signal goes from "0" to "+" . . .	13
3.1	Subtasks and States of the Stacking Task.	18
4.1	Vectors of Force and Velocity Along Trajectory of Motion.	27
4.2	Screws Describing Constrained Motion During Needle Insertion by a Planar Manipulator.	31
4.3	Instantaneous Center of Rotation	33
5.1	(a) Master Manipulator with Operator's Hand. (b) Remote Manipulator. .	36
5.2	Horizontal and Vertical Positions of the Two Fingers. Legend: m = Move- ment of fingers to a new position; gg = Grasping an object on the ground; g1 = Grasping object 1; g2 = grasping object 2.	41

5.3	Mean (a) Horizontal and (b) Vertical Velocities of the Two Fingers. Legend: m1 = Move block 1 to a new position; m2 = Move fingers back to get block 2; m3 = Move block 2 to a new position.	42
5.4	Forces: (a) Total Horizontal Force on Environment; (b) Total Vertical Force on Environment; (c) Grip Force; (d) Vertical Shear Force. Legend: g1 = Grasping object 1; g2 = Grasping object 2.	43
5.5	Result of Automatic System State Identification Procedure.	44
5.6	Object Weights. The solid portions are the segments of data used for weight estimation.	48
5.7	Object Heights. The solid portions are the segments of data used for height estimation. Object 2 height is the difference between the first and second solid lines, while object 1 height is the difference between the third and fourth solid lines.	48
5.8	Object Widths. The two solid lines indicate the widths of the blocks. . . .	48
5.9	Lower Bound of μ , Vertical Force F_{y3} , and Horizontal Force F_{x3} . μ is only calculated during "hold on ground" and "hold above ground" states. . . .	49
5.10	Position of Crank Handle, Vectors of Applied Force, and Computed Center of Rotation. The vectors show direction and magnitude of force applied by the robot hand.	51
5.11	Tangential Force, F_t , for the Crank and the Least Squares Approximation Given by $mg = 0.16$ N, $\mu = 0.053$ and $c = 0.010$	52
5.12	Constraint Forces, F_C , for the Block and Crank.	53

5.13	Constraint Force and Displacement in Insertion Direction. a = free motion, b = insertion, c = penetration, d = extraction.	54
5.14	Model of History Dependent Stiffness. The position of the needle tip corre- sponding to each number on the plot shows configuration information at key transition points in the insertion process.	55
5.15	Block Diagram Showing Interactions Between the Operator and the Virtual Environment	57
5.16	Master and Remote Manipulators (left) and Close-up of Remote Manipulator (right) During Block Calibration.	58
5.17	Virtual Simulation of Calibrated Block.	58
5.18	(a) Horizontal Forces Acting on Virtual Block. (b) Vertical Forces Acting on Virtual Block.	60
5.19	(a) Grip Force Acting on Virtual Block. (b) Grip Shear (vertical) Force Acting on Virtual Block.	61
5.20	Virtual Block Weight Measurement. The solid portions are the intervals during which the virtual block is being held above ground.	61
5.21	Virtual Block Width Measurement. The solid portions correspond to the intervals during which the virtual block is held above ground.	62
5.22	Position of Virtual Crank Handle, Vectors of Applied Force, and Computed Center of rotation. The vectors show the direction and magnitude of force applied by the virtual fingers on the virtual crank.	63

Chapter 1

Introduction

This thesis presents a method to automatically estimate parameters of remote environments in a teleoperated robot system. These parameters can then be used to assist the operator with task performance or to calibrate virtual models of the remote environment. Use of the term “automatic” means the parameter identification procedure is performed by the estimation system with little or no input from the human operator. The method could be implemented so it also runs completely autonomously. However, for the implementation described in this thesis, each portion of the method was performed individually and then results from each were concatenated by hand to obtain the final result.

Figure 1.1 depicts the concept of the estimation system. As a normal teleoperated task routine is performed, the system collects data about the task being performed, including task descriptions, operator commands, and the resulting forces and motions in the remote environment. It is assumed that there are appropriate position and force sensors available to measure such information. The estimator then analyzes this information to determine an appropriate model of the remote environment including parameters describing object

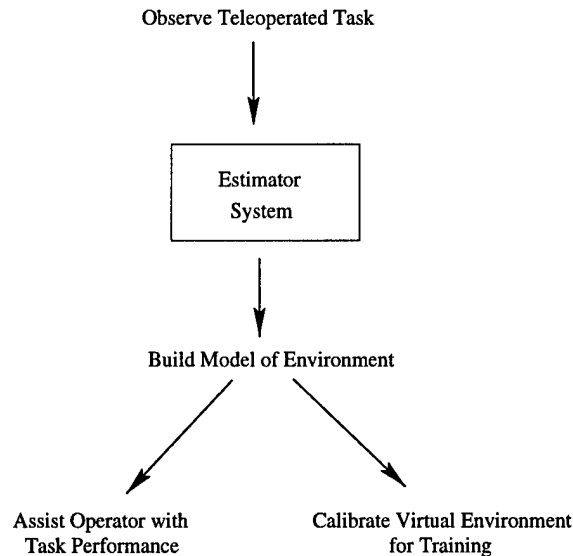


Figure 1.1: *Flow Diagram of the Parameter Identification Problem.*

properties and kinematic constraints.

The model can then be used for two applications. The first is to assist the operator with the task. This “assistant” can provide the operator with real time or near real time data on objects in the remote environment. It can alert the operator to impending collisions and excessive forces being applied, or it can be used to provide information on object size, type of contents contained within the object, etc. The second application of the model is for use in calibration of a virtual environment to closely resemble the real environment. Such a virtual environment can be used in designing training simulators for operator trainees. The advantage of using this method for the calibration is that it relies on actual data from a real system rather than best guesses of what a real system might be like, resulting in a more accurate virtual model and less time spent in calibration.

The estimation system determines two types of parameters. The first type is remote environment object properties such as object mass, moment of inertia, size, stiffness, and

friction coefficients. These can be used to infer information about the object being manipulated. For example, if the object is a barrel, density can be used to infer contents of the barrel. A changing center of inertia could be used to determine whether the contents are liquid or solid, while barrel structural integrity can be inferred from stiffness. The second type of parameter is a constraint parameter. Constraints determine allowable directions of motions of objects. An example of a constraint is a cable tied around a barrel anchoring it to the ground. The cable acts a constraint limiting the barrel's motion.

1.1 Application Examples

By considering several examples one can see how the "assistant" increases task performance speed and enhances safety, and how the creation of a virtual environment allows for design of realistic training systems. Take the realistic case of Toxic Waste Remediation, or the similar task of Explosive Ordinance Disposal where teleoperation allows for the operator to work at a safe distance. While the toxic waste barrels are being moved, the estimator can provide information on barrel size, contents, wall strength, and any constraints to its motion [11]. The operator can then use this information to determine how best to approach the removal task, how much extra caution to take, if there are any impediments to movement of the barrel, and what the disposition of the barrel should be.

Furthermore, the information determined by the estimator can be used to create a virtual training system, whereby novice operators can learn how to do the job and practice difficult tasks without risk of damage to equipment or damage to objects in the remote environment. Referring again to the example, the barrels in the virtual trainer would have

the same dimensions impose the same reaction forces as those in the real remote environment because they were created and calibrated using information determined by the estimator from real data. Trainees would reap the benefit of a true task performance experience without risking toxic waste leakage resulting from puncture of containers or damage of excavating machinery.

Surgical training systems are another example where such a training system would be highly beneficial [32, 33]. By observing a real surgical procedure, the estimator would be able to determine parameters for calibrating a surgical simulator. The simulator could then be used by medical students to practice the procedure and gain vital experience without ever having to risk making a mistake on a human patient.

For cases where the distance between operator and remote environment are great, such as a Mars expedition to collect soil samples, time delays due to signal travel time impede task performance. The estimation system could be used to create a virtual model of the remote environment in which the operator would perform a task in what appears to be near real time [27, 4, 9]. As the task is performed, the virtual model would be continually updated by the interactions of the actual remote manipulator interactions with the planet's surface.

Another example where this estimation system could be implemented is military aircraft munitions loading [19]. This example, just like each of the previous ones, demonstrates a benefit both from the task assistant application and the virtual training system application.

1.2 Background

Teleoperation is a process in which a human operator at one location manipulates objects in some remote location. The top portion of Fig. 1.2 shows the interactions between the operator and the remote environment of a typical teleoperated system. To perform a task, the operator inputs a command by manipulating a controller interface. This interface can be a joystick, a set of levers and switches, an instrumented data glove, etc. As the operator moves the interface, force and position sensors detect this motion and the system sends a corresponding signal to motors on the remote manipulator. The motors then move the manipulator to the position commanded by the operator. Information about motions and forces in the environment can then be fed back to the operator, allowing him or her to make more informed decisions on how to manipulate the controller to effect the desired changes. This feedback can be visual, auditory, and/or force feedback. Force feedback means that forces acting on the remote environment are measured by sensors on the manipulator, and corresponding forces are imparted through the controller interface to the user's hands. In this way, the operator feels what's happening, as well as seeing and hearing it.

When a property estimator is included in the feedback loop, interactions occur between the operator, remote manipulator, and the estimator (See the remainder of Fig. 1.2). Now the estimator receives as inputs, information about the operator's movements, reactions in the remote environment, and the task description. At the same time, information is fed back to the operator to improve speed, safety, and quality of task performance. While in the ideal case, the data collection for parameter estimation would be transparent to the operator, there may be cases in which the data set collected may not be rich enough for

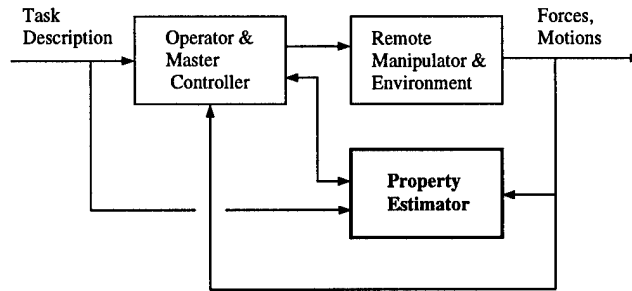


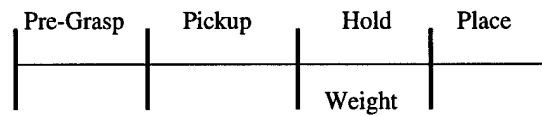
Figure 1.2: *Block Diagram of a Teleoperation System with Property Estimator.*

the estimator to provide accurate information to the operator. In these cases the estimator would need to ask the operator for additional task description information and/or repeated performance of the task to provide the richer data set.

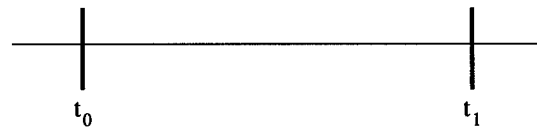
1.2.1 Three Sub-Problems of the Environment Identification Problem

Given the inputs to the estimator, the environment identification problem is composed of three main sub-problems: task decomposition, data segmentation, and parameter estimation [34]. Figure 1.3 depicts this process with the example of a simple pick and place task. The first step, task decomposition, involves decomposing the task into its constituent sub-tasks. This is necessary because each property of interest can be determined only during the occurrence of certain subtasks. In this case the subtasks are *grasp*, when the manipulator makes contact with the object; *pickup*, when the object is lifted from the ground; *hold*, when the object is held above the ground; and *place* when the object is put back down on the ground. The property of interest is weight and it can only be determined during the *hold* subtask. The subtasks are defined in terms of the configuration of contact states. In order to concentrate on the parameter identification problem, it is assumed in this thesis that the task decomposition into subtasks is given.

1. Specification of Subtasks



2. Identify Segments in Datastream.



3. Estimate Object Properties and Kinematic Constraints

$$\text{Weight} = \frac{\sum_{t=t_0}^{t_1} F_y(t)}{t_1 + t_0 - 1}$$

Figure 1.3: *The Three Parts of the Environment Identification Process Applied to a Pick-and-Place Task.*

The next step, data segmentation, is the determination of the time intervals during which each of the subtasks occur. This will specify the data points to be used in estimating the parameters. For property estimation, segmentation may not require classification of every portion of the data stream.

Once the data is segmented in time, parameter estimation can proceed using sensory data within each of the segments, using techniques appropriate for that type of parameter. For object properties, it can take the form of an equation applied to the data as shown in Fig. 1.3.

Kinematic constraints can be identified as well. Constraint identification involves determination of whether a constraint exists, modeling of the constraint, and identifying the associated parameters. Existence of a constraint is determined by examining the forces acting on an object to see if there are additional forces aside from the expected forces such as weight, friction, etc. A model of the constraint must then be developed to describe constraint forces and the motion limitations imposed by the constraint. Finally, once the model is determined, parameters to fit in the model are estimated using the same methods used for determining object properties. The details of the methods used in parameter and constraint identification will be explained in detail in chapters 3 and 4.

1.3 New Work and Thesis Outline

This thesis presents for the first time the concepts of

- Automatic parameter identification in a teleoperated system. While previous work on the sub-problems of the parameter identification process exists, work presented

here for the first time combines these sub-problems to achieve the desired automatic parameter identification.

- Automatic identification of constraints which includes, determination of existence of constraints, modeling of constraints, and determination of constraint parameters. In previous work, a model of the constraint was assumed to be known and the parameters to fit the model were a combination of known and unknown values. Here, no prior knowledge of the constraint is used in developing the model.
- Automatic calibration of virtual models. Previously, when virtual models were created, the values of model parameters were chosen using a best guess. The values were then refined by an iterative process until the model provided the correct haptic perception to the user. Here, actual data from a real task is used to calibrate the virtual model of that task.

The remainder of this thesis begins by reviewing the literature on previous work in this field in chapter 2, and continues with a discussion of the theory of parameter identification in chapter 3. Chapter 4 discusses the theory of kinematic constraint identification. Chapter 5 presents experiments that were performed to demonstrate the application of the theory and the results of those experiments. Chapter 6 concludes with a discussion on the implications of these results and then presents areas of further research.

Chapter 2

Literature Review

This chapter describes prior work on the individual sub-problems of property estimation. A limited amount of work on task decomposition is first described. Next, methods of data segmentation are discussed with an emphasis on Qualitative Reasoning and Hidden Markov Models. While Parameter Estimation has a rich literature, only those papers applicable to the systems of interest are reviewed. Finally, the limited literature relating to calibration of haptic feedback systems is discussed.

2.1 Task Decomposition

While not extensively studied, task decomposition has been investigated from a variety of viewpoints according to the goal of the decomposition. The earliest work known to the author is that of Kondoleon [17] who analyzed ten common products and found they could all be assembled using twelve manufacturing sub-tasks. He did this to find the statistical occurrence of each task in assembly operations. Nevins and Whitney [26] used the task

decomposition approach of Kondoleon to estimate assembly cost. They accomplished this by finding the cost of performing each sub-task and the frequency at which it occurred in an assembly operation.

More recent work on task decomposition has been motivated by an interest in subtask-specific control strategies. In this context there were two concerns. The first was to discern the control strategy employed in a successfully completed sub-task by examining task data. An example of this was McCarragher's work [23] where he defined states of contact vs. non-contact in trying to understand force signals generated by a human subject.

The second concern was where control laws were designed for particular subtasks and transitions had to be detected on-line in order to switch control laws. To do this, Cutkosky and Hyde [5] divided a task into phases separated by key transition events. McCarragher and Asada [25] used states of contact for recognition of state transitions of assembly processes.

In none of these cases, however, did the authors make a differentiation between a *state* which is defined as the "alphabet" or "building blocks" of a given task, and a *subtask* defined as the task specific ordering of *states* as used in this research.

2.2 Segmentation

This section will discuss the goal of various segmentation algorithms in the literature and then discuss the methods used to segment data.

The transference of human skills to autonomous robots is one way to develop control algorithms for the robot task. In order to do so, researchers found it necessary to segment the data. For example, in order to understand the qualitative control characteristics of

an example task performed on a teleoperated system, Pook and Ballard [29] segmented data from a teleoperated task. Kang and Ikeuchi [16] were interested in assembly task programming and used segmented data from a grasp task for the purpose of understanding the grasp motions. Likewise, Yang et al. [36] segmented data for similar purposes. Delson and West [6] used human demonstration to program robots and in the process had to segment the data into subtasks that facilitated the generation of a robot program.

Segmentation methods described in the literature include Hidden Markov Models [12], Qualitative Reasoning with Thresholding [23], Neural Networks for off-line segmentation [8], and Petri Nets [24]. The first two methods listed dominate the literature and are described here.

2.2.1 Qualitative Reasoning with Thresholding

McCarragher [23] demonstrated a qualitative reasoning approach to analyze the sensor signals from human task performance and identify changes in contact state. In the first stage, the measured motions and forces at the finger tips are thresholded. The resulting threshold functions are tri-valued, that is, they are assigned values of “+”, “0”, or “-”. Figure 2.1 demonstrates how this works. The solid line is an example of raw data which could represent a force signal. When the signal is below some threshold value, the threshold function indicates a “0” value. As soon as the force signal exceeds some threshold value, the resulting threshold signal represented by the dotted line jumps to a “+”. Similarly, if the signal goes below a set negative threshold, the threshold signal would jump to a “-”.

In the second stage, thresholded data is used to develop a qualitative understanding of the motions and forces associated with particular contact states. Sets of logical rules,

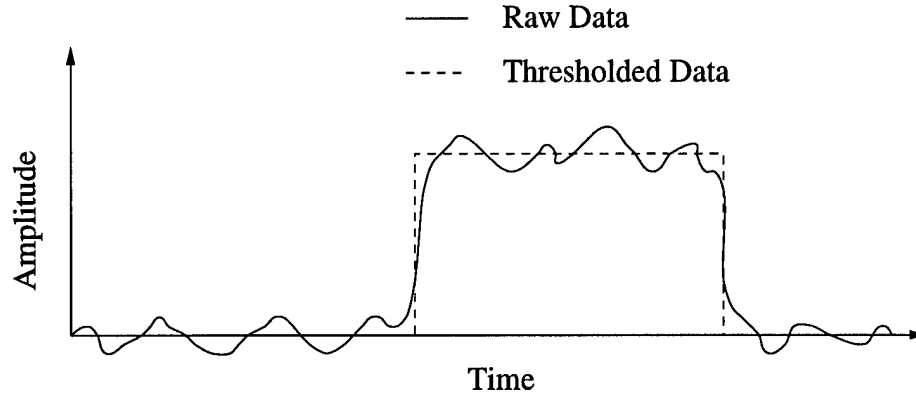


Figure 2.1: *Example of the Thresholding Technique. When the raw data exceeds a given threshold value, the value of the resulting signal goes from “0” to “+”*

using Boolean operators, are developed which encode this understanding of contact states. For example, if the data in Fig. 2.1 represented finger force directed normal to a constraint surface, a thresholded signal of “0” indicates that no contact has been made. The transition to a signal of “+” indicates that contact has been made. More generally, thresholded values of positions, velocities, and forces can be combined in Boolean expressions to determine the current state from a variety of possible states.

2.2.2 Hidden Markov Model

A Hidden Markov Model (HMM) is used to determine an unknown sequence of system states based on another set of related data that is known. The HMM models the relationship so that by analyzing the known data set it can predict or estimate what the corresponding state sequence must be. Hannaford and Lee [12] applied the method of using HMM’s from applications in speech processing [31, 30] to the analysis of teleoperator force and torque

data for the purpose of task segmentation. This section describes how the HMM works and how it is developed.

As an example of how HMM's work, we look an example described by Rabiner [30]. Think of a set of three urns hidden behind a curtain. Each urn contains many marbles of several different colors. A process occurs by which a marble is pulled from one of the urns and then brought out in front of the curtain where it can be seen in full view revealing its color. This process is repeated several times and the color of the marble is recorded each time resulting in a data set made up of a sequence of colors. An HMM can be used to determine the order of the urns from which the marbles were drawn. The answer is by using an HMM. In HMM terminology, the urn from which the marble is drawn is the state of the process at that particular time. The HMM models the process and can be used to determine the unknown sequence of urns (sequence of states) based on knowledge of the marble color order (known data set).

In teleoperation applications, the manipulator task is viewed as a finite state process in which states correspond to sub-tasks. A Markov model is used to represent the sequence of task states. It is referred to as a Hidden Markov Model since the estimator cannot view the sequence of states directly. Only the observations consisting of the force and motion data streams are available.

The HMM, γ , is described by $\gamma = \gamma(A, B, \pi)$ where

- A is the probability density function of being in some state given the previous state.
- B is the probability density function that a given state results in a given observation.
- π is the initial state probability distribution.

Three main problems involving HMM's are as follows:

1. Given an observation sequence or set of sequences, determine the locally optimal HMM.
2. Given an HMM and an observation sequence, what is the most likely sequence of states?
3. Given an HMM, what is the probability of occurrence of some observation sequence?

The three problems can be solved using the Baum-Welch method, the Viterbi algorithm, and Forward-Backward method, respectively [12, 30, 31]. Problem 1 must be solved first to determine the HMM appropriate to a task and set of training data. Problem 2 is solved subsequently (either on-line or off-line) to segment the data stream. Problem 3 provides a measure of accuracy of the chosen model and can be used to evaluate the accuracy of the state sequence determination.

The advantage of using an HMM over other methods for segmentation is that it is more robust and does not have problems handling oscillations in the data or changes in the state sequence. The advantage of using qualitative reasoning, though, is that it requires less computation and no extensive model development.

2.3 Parameter Estimation

Extensive work has been done on parameter estimation [21]. A portion of this work is devoted to robot parameter identification. For example, the identification of link inertial parameters has been studied by Khosla and Kanade [18] and An et al. [2]. Others have

investigated the identification of kinematic parameters [7]. In addition, a few authors have addressed identification of robot payload and environment properties. Methods for estimating payload inertia appear in the work of Atkeson et al. [3] and Lin and Yae [20]. Lin and Yae also estimate certain parameters relating to constraints of the operating environment. They do assume, however, that the constraint surface is at least partially known and that it can be modeled using a set of known and unknown parameters. Still others have studied how to determine shapes of objects in the operating environment by using haptic exploration [1, 28].

2.4 Calibration of Haptic Feedback Systems

Current methods used to create virtual models are usually based on idealized assumptions regarding the physics of object interactions. The parameters of these models (i.e., mass, stiffness, viscous damping coefficient, etc.) are not always well known. Often their values are selected arbitrarily and then adjusted iteratively until the virtual model “feels” like the real object. This process is obviously subjective, and without knowledge of the actual parameter values, the tuning process could lead to a locally optimal solution far from the actual one.

The method proposed in this thesis is superior in that actual data is used to achieve automatic calibration of the model. This notion has been explored concurrently in a simplified fashion by MacLean [22]. In her work, force-displacement data from a toggle switch was recorded and used to recreate the haptic sensations for a virtual switch.

Chapter 3

Theory of Parameter Identification

This chapter describes in detail the implemented solutions to the three sub-problems of the estimator. As described, it is assumed that models of the object interactions are provided. Chapter 4 on kinematic constraints addresses situations where models of the object interactions are not known a priori.

3.1 Task Decomposition

Task decomposition is the process of breaking down a task into *subtasks* associated with the particular properties we are interested in determining. Associated with each subtask is a *system state* that describes the physical configuration of the robot and work space objects. While in previous work, subtask and state were used interchangeably, a distinction is made here. States are the “alphabet” or “building blocks” of a task which describe the possible physical configurations. Subtasks are a task-specific ordering of states of the robot and objects. Each state can correspond to more than one subtask since physical configurations

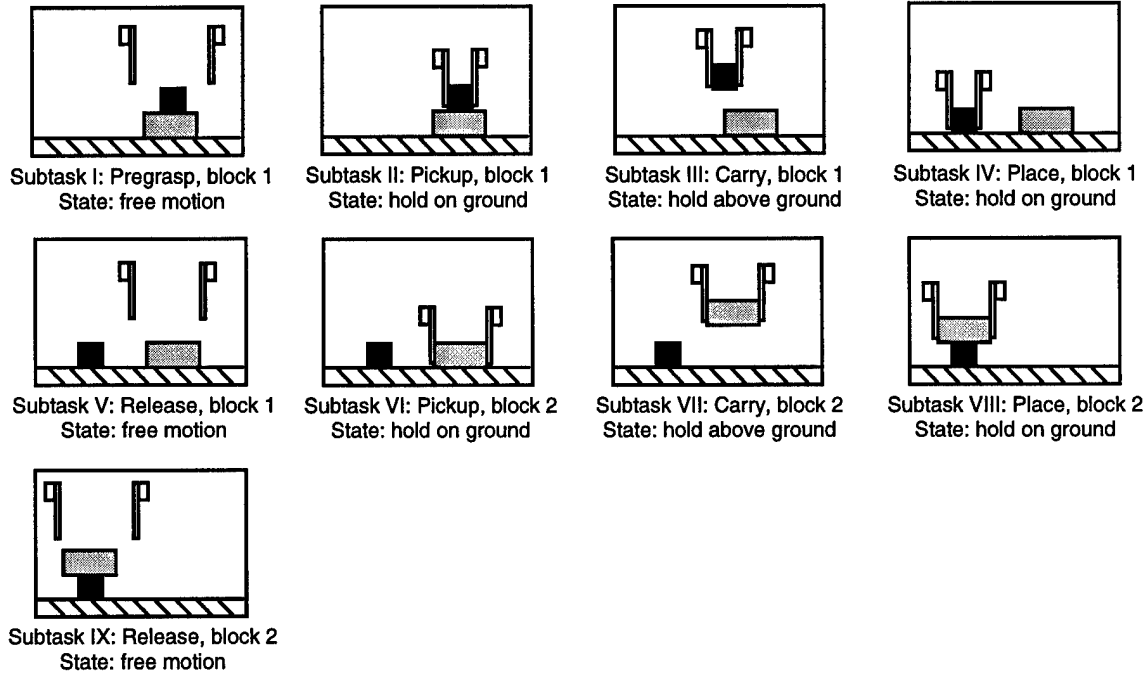


Figure 3.1: *Subtasks and States of the Stacking Task.*

can be repeated as the task progresses but each subtask occurs only once.

The block stacking task shown in Fig. 3.1 illustrates the concepts of *states* and *subtasks*. In this task, a stack of two blocks is unstacked and re-stacked using the a two-fingered manipulator from a teleoperated system. The task begins with Subtask I, *pregrasp of block 1*, followed by subtasks *pickup*, *carry*, *place*, and *release* of the first block. The ensuing subtasks for the second block are *pickup*, *carry*, *place*, and *release*.

Each subtask in Fig. 3.1 is also labeled with a system state, which describes the configuration of the manipulator robot and environment at a specific time, including contact and constraint conditions. Three states are shown: *free-motion* of the fingers, *hold-on-ground*, and *hold-above-ground*. A fourth possible state, *hold-on-ground-with-sliding*, is not shown in Fig. 3.1, although it could occur during any *hold-on-ground* state.

Knowledge of both state and subtask are important to the identification of object properties. The state determines which properties can be estimated. For example, the weight of an object might be best estimated during a state in which the object is held above the ground. Knowledge of the subtask is also needed in order to determine which object is currently being grasped.

In this study, it is assumed that the parameter identification procedure begins after the task is complete, so that the entire position, velocity, and force records are available. Also pre-specified are the expected sequence of subtasks and their corresponding states for each task. The correspondence between states and the parameters which can be identified in that state is also given explicitly. However, this information can be compiled in a database for use with a variety of tasks; for instance, the weight of a grasped object can be determined in every *hold-above-ground state*. Knowledge of the subtask order is used to relate the identified parameters with particular objects in the workspace.

3.2 Segmentation

Segmentation is the process of breaking up the data stream into time intervals that correspond to different system states. As mentioned in the previous chapter, there are several segmentation methods that can be used including Qualitative Reasoning with Thresholding, Hidden Markov Models (HMM's), Petri Nets, and Neural Networks. The qualitative reasoning technique was used here because it is both easy to implement and fairly robust.

Segmentation of the data stream into intervals corresponding to the task states is accomplished using a modified version of McCarragher's method [23]. The modification provides

two important improvements. The first is to analyze pertinent task motions and forces rather than raw force and position data providing better discernment of the overall configuration changes. The second improvement is the differentiation between subtasks and states allowing a discernment between different objects that are manipulated.

To accomplish these improvements, McCarragher's two stage method is modified into three stages as follows: In his first stage, McCarragher thresholded the data to three possible qualitative values of "+", "0", and "-". This stage is modified by first transforming the measured motions and forces at the finger tips into task motions and net forces, and then thresholding the transformed data. McCarragher's second stage in which the thresholded data is combined using a set of logical rules to assign state labels remains the same. In the added third stage, the subtasks are identified. The following describes the details of these stages.

3.2.1 Stage 1

For the first stage, it is necessary to first transform the data into pertinent task motions and net forces. Looking back to the block stacking task illustrated in Fig. 3.1, it should be noted that there are two fingers grasping the blocks. Because tasks are often performed using a multi-fingered manipulator, the finger tip motion and force data of those fingers must be transformed to find the net motion and force of the grasped object in task space. This requires computation of averages and differences of the horizontal and vertical components of the two finger motions. The average motion of the two fingers is a measure of rigid body motion, and the difference in finger tip motions corresponds to gripping motion. Horizontal and vertical components of the finger tip forces are similarly transformed. The sum of the

forces from the two fingers is the net force applied to the environment (including the object and any surface it contacts such as the ground), and grip force is the minimum horizontal component of the two (opposing) finger tip forces.

Once the task motions and forces are obtained, the velocity and force data are passed through thresholding filters with equal positive and negative thresholds. These threshold values can be chosen without much difficulty based on the range of values seen in the data. The threshold value should be chosen so noise does not cause the signal value to surpass the threshold value but changes in the data due to task motions and forces do.

3.2.2 Stage 2

In the second stage, a set of formalized logical rules is applied to determine the sequence of the system states. These rules are based on expected task motions and forces of the given task. For the example of the block stacking task previously discussed, these rules, as summarized in Table 3.1, would be as follows: *Free-motion* is defined as any time that both components of force on both finger tips are zero. The *hold-on-ground* state is active whenever the grasp force is positive (i.e., greater than the positive threshold) and the total vertical force exerted by the fingers is not upward (i.e., negative or zero), the average vertical velocity of the fingers is near zero. The *sliding-on-ground* state is the same as *hold-on-ground*, with the added condition that the average horizontal velocity is non-zero. *Hold-above-ground* is active when the grasp force is positive and the sum of the vertical forces is upward (positive). Note that this criterion for the hold-above-ground state can include a brief period just before and after the block lifts when the vertical force is positive but the block is still in contact with the ground. This is unavoidable if the state identification

Table 3.1: *Qualitative Values of Sensed Parameters Which Define Task States*

State	Sensed Parameters					
	Velocity Horizontal	Velocity Vertical	Average Horizontal Force	Average Vertical Force	Grip Force	Shear Force
<i>free-motion</i>			0	0	0	0
<i>hold-on-ground</i>		0		– or 0	+	
<i>hold-on-ground, sliding</i>	+ or –	0		– or 0	+	
<i>hold-above-ground</i>				+	+	

algorithm uses only instantaneous data without reference to the prior state of the system.

3.2.3 Stage 3

In the third stage, subtasks are associated with each identified state. This is done using a priori knowledge of the expected sequence of subtasks as determined during task decomposition, and additional historical information. The sequence defines that certain subtasks cannot occur until others have taken place. For the example task where block 1 is picked up and placed back down followed by block 2 being picked up and replaced, the subtasks *carry-block-1* and *carry-block-2* both occurred during a *hold-above-ground state*; however, they are separated by a *free-motion state*, a fact used to distinguish the two subtasks.

Several issues complicate the state and subtask identifications of stages two and three. For example, oscillations in the signal due to contacts not being made and broken cleanly can create time intervals during which no state is identified. One solution is to mark these time intervals with a *no state* tag so they are not used in the parameter calculations. Another problem is that because of these unclean contact breaks there is potential for oscillation between two states. Looking at the pick and place task again, this could result in an

oscillation between the *hold-on-ground* and *hold-above-ground* states. The solution here is to avoid estimating parameters during intervals in which states are rapidly oscillating.

3.3 Parameter Estimation

Given an equation describing the physics of robot-object interactions in a particular contact configuration and a portion of the data stream for which this equation is valid, the problem is reduced to that of parameter estimation. In this thesis, only linear, time-invariant parameters are considered and so a least squares method is sufficient.

In a typical case, we have the following:

$$[A][x] = [b] \quad (3.1)$$

where A is the input data set, b is the output data set and x is the set of parameters to be estimated. When using real data, this usually results in an over-determined set of equations. The least squares solution for x employs the pseudo-inverse matrix [35]:

$$A^+ = Q_2 \Sigma^+ Q_1^T \quad (3.2)$$

where the columns of Q_1 are the eigenvectors of AA^T and the columns of Q_2 are the eigenvectors of $A^T A$. The r singular values, $\sigma_1, \dots, \sigma_r$, on the diagonal of Σ are the square roots of the nonzero eigenvalues of both AA^T and $A^T A$, and the reciprocals, $1/\sigma_1, \dots, 1/\sigma_r$, are on the diagonal of Σ^+ .

A^+ minimizes the mean squared error of the parameters. The optimal solution of the parameters x is

$$[x^+] = [A^+][b] \quad (3.3)$$

3.3.1 Estimation Difficulties

Several difficulties arise during parameter estimation. The first is oscillation in the data signal used in computing a parameter. This situation occurs most frequently at the beginning and end of subtask because of intermittent contact during state transitions. Recognizing that most of the oscillation occurs at the beginning and end of a subtask, a solution is to discard a percentage of data at both ends of the interval.

A second problem is the determination of parameters that are non-linear or time varying. An example is a punctured barrel leaking its contents. With the possibility of time-varying parameters, the question arises as to whether the change is due to an actual parameter change or to error drift in the data collection system.

Chapter 4

Constraint Identification and Modeling

Earlier it was assumed that a dynamic model describing interactions of the robot and workspace was provided for each state of interest. In this case, any constraints present in the system are defined implicitly by the dynamic equations. However, additional constraints may be encountered while moving objects in the workspace. In these situations, it is desirable for the estimator to both detect the existence of a constraint and to model it based on the data stream.

This chapter describes the theory of constraint identification and modeling. In the first section, constraints are defined and a method for their identification is presented. Several techniques for modeling constraints are then presented and compared.

4.1 What is a Constraint?

A constraint prevents or opposes motion in one or more directions. Defining a constraint involves finding the allowable displacements and associated constraint forces at every possible configuration of the system.

A constraint can be thought of as either rigid or non-rigid. In the real world, there is no perfectly rigid constraint. Every surface has some stiffness associated with it, albeit an extremely high stiffness in some cases. For the purposes of this thesis, a distinction is made in that if the stiffness exceeds a set value it is considered a rigid constraint. Otherwise, it is considered a non-rigid constraint. This maximum value is a function of the inherent stiffness of the manipulator system and the fidelity of the sensing system.

A rigid kinematic constraint reduces the number of degrees of freedom of an object. An example of this type of constraint is the effect of the rail on a mono-rail car. The rail constrains the car to move only in a direction of motion parallel to the rail. Figure 4.1 shows the trajectory of such an object and the force vector on the object at one point along the path. If this object is undergoing pure constrained motion, then the component of the force normal to the direction of the instantaneous velocity v , F_n , is due to the constraint. The force in the direction of v , F_t , is due to other task-related interactions, such as frictional forces from sliding on the constraint surface. This suggests that a basic method of identifying constraint forces and separating other types of forces from them is decomposing the net force into components normal and parallel to the direction of motion.

In the case of a non-rigid constraint, the constraint acts as a force that opposes motion in a given direction but does not prevent it. Consider, as an example, a trampoline surface.

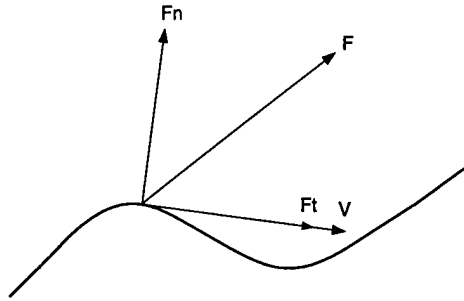


Figure 4.1: *Vectors of Force and Velocity Along Trajectory of Motion.*

As an object moves normal to the surface, it encounters a resistance to this motion which in this case becomes greater the farther into the surface the object tries to move. Since the force is dependent on displacement it is a configuration-dependent constraint.

A non-rigid constraint will depend on the compliance of both the manipulator and the object. If the object is much stiffer than the fingertip, it is difficult to find the object stiffness because it can be lost in the scatter of data of the less stiff fingertip. When the stiffness of the fingers becomes equal to or greater than that of the object, object stiffness may be determined with more confidence.

Motion in the constrained as well as the unconstrained directions can be both configuration and history dependent. For example, under the Coulomb friction model, static friction may prevent an object from sliding until the friction force is exceeded. Before sliding, the object is completely constrained and no motion occurs. However, once sliding begins, a former constraint direction becomes a direction of motion. As another example, consider the insertion of a needle through tissue and into a body cavity. Before contact with the tissue, motion is unconstrained. After contact, the insertion motion requires a force dependent on

the insertion depth as tissues of different mechanical properties are traversed. When the body cavity is penetrated, the resistive force decreases abruptly. Finally, the force required to extract the needle may be configuration dependent as well.

The conclusion is that any general approach to modeling non-rigid constraints must incorporate both configuration and history dependence for such properties as stiffness and damping. In addition, these properties may be nonlinear and only piecewise continuous.

4.2 Identifying the Presence of a Constraint

This investigation of constraint identification focuses on the experimental data presented in chapter 5 where robot fingers grasp an object and execute a task. Our method for establishing the existence of kinematic constraints on the manipulated object requires the identification of the forces of constraint. In order to do so, the following assumptions are made:

- The forces and torques applied to the manipulated object by the fingers are known.
- Knowledge of the finger trajectories (and the fingertip compliance) permits computation of the object trajectory.
- Task information specifies the major contributors to the net forces and torques applied to the manipulated object.

The approach to identifying constraint forces is as follows. By the first two assumptions, applied forces and torques are projected along the tangential and normal directions of motion obtained from velocity data. Based on the third assumption, dynamic equations

are written in terms of the normal and tangential motion coordinates. Any remaining unexplained forces or torques normal to the direction of motion are attributed to constraints. Occasionally, tasks involve the penetration of constraints, e.g., needle insertion. In these cases, the constraint is identified by unexplained forces or torques in the tangential direction. Data from repeated trials provide a means to refine force and displacement relationships and to determine if constraints are unilateral or bilateral. The examples in chapter 5 will clarify this procedure.

4.3 Modeling of Constraints

Once the existence of a constraint has been established, it must then be modeled. The model or representation must accurately describe the allowable trajectories of motion and the accompanying forces, if any, at each position. This section discusses two techniques for modeling motion constraints. In the first technique, splines are fitted to the motion data. The majority of effort was devoted, however, to the development of the second technique based on screw theory. Modeling the center of rotation is presented as a special case of this approach.

4.3.1 Spline Representation

One way to represent allowable trajectories is to find an equation describing the path of the trajectory. To do this one can use curve fitting techniques to get the equation of a curve that matches the trajectory of allowable motions. Simply applying a polynomial fit can result in an equation of high order, and over large intervals, it can vary greatly from the actual data. Instead, a spline fit can be used which overcomes these problems by approximating

the trajectory function with a smoothly continuous series of piecewise low order functions. Once an equation of the trajectory is found, then the forces at each point must be recorded for a complete representation of the system.

4.3.2 Joint / Screw Representation

Constraints can be expressed as joints with associated configuration-dependent stiffness, damping, friction, etc. Recall from kinematics that multiple degree of freedom joints can be composed from joints which each possess a single degree of freedom. The three single degree of freedom joints are the revolute, prismatic and screw joints. Furthermore, screws with zero and infinite pitch (i.e., the ratio of translational to rotational motion) can be used to represent revolute and prismatic joints, respectively. This generality is exploited in screw theory in which the geometry of the screw is used as the basis for describing allowable instantaneous motion of a mechanism.

A *screw* is defined by its axis, a line in \mathcal{R}^3 , and by its pitch, which has units of length. In applying screw theory to kinematic chains composed of rigid links, the usual practice is to describe the instantaneous motion of a particular link as a linear combination of the configuration-dependent screws associated with the mechanism's joints. In this way, constraints do not explicitly appear in the set of screws or *screw system* describing the allowable motion. In other words, each additional constraint means one less screw direction.

Some readers may be familiar with wrenches which are coaxial force and moments acting along and about a screw axis. Wrenches are not used explicitly here since constraints are represented as joints with configuration-dependent forces. In particular, constraints are modeled as follows.

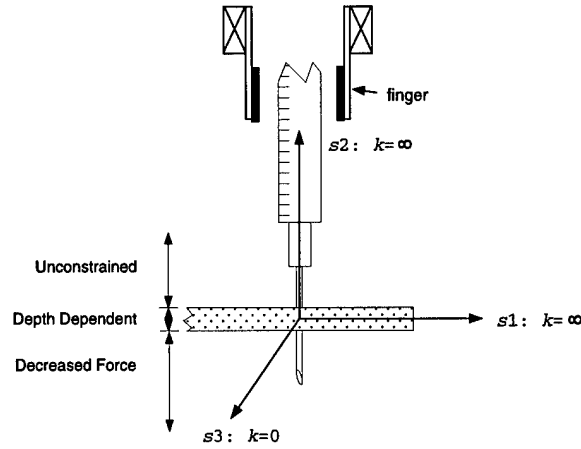


Figure 4.2: *Screws Describing Constrained Motion During Needle Insertion by a Planar Manipulator.*

1. *Constraints which are both rigid in comparison to the manipulator and configuration independent are modeled in the usual way by screws describing the allowable motion.*
2. *Constraints not meeting these conditions are modeled by screws in the constraint direction(s) which possess configuration-dependent properties, e.g., stiffness and damping.*

As an example of how the screw representation is implemented, consider the insertion of a needle into tissue by a planar manipulator as shown in Fig. 4.2. This example is particularly interesting as the primary motion involves the penetration of a constraint surface. Only screws associated with the three planar degrees of freedom are shown since the constraints associated with out of plane motion are considered rigid.

Screws s_1 and s_2 , in the horizontal and vertical directions, respectively, are of infinite pitch ($k = \infty$) and represent allowable translations. Screw s_3 is of zero pitch ($k = 0$) and represents rotation in the plane of the page. For any configuration of the needle in which contact is not made with the tissue, there are no constraint forces associated with the screws. As the needle penetrates the tissue, constraint forces and torque, which are

dependent on penetration depth, can be associated with s_1 , s_2 and s_3 .

4.3.3 Center of Rotation Representation

A Center of Rotation (COR) representation describes the trajectory of motion by tracking the instantaneous COR at all time points and the corresponding radius which is the distance from the COR to the trajectory. The constraint forces are then described by tracking the measured force at each COR and radius. This representation is actually a specific case of the screw joint representation described in section 4.3.2.

For motion along any trajectory, an Instantaneous Center of Rotation (ICOR) can be found. The ICOR is defined as the point of intersection of the normals to the direction of motion at two points along a trajectory (See Fig. 4.3). In the limiting case of motion along a straight line the ICOR is at infinity. For the case of a circular trajectory, the ICOR is always at the center of the circle. While the ICOR should be a single point for a circular trajectory, the affect of finger slip caused the ICOR to be less than smooth. This effect can be reduced by using data averaging techniques.

For each point of this calculated ICOR, one can then find the associated radius and forces. The radius can be found using trajectory position with respect to the COR. The direction and magnitude of the corresponding forces are also recorded. This results in a complete representation of the constraint.

4.4 Issues in Determining the Best Representation

Visual feedback is an important component of any actual or simulated teleoperation system. Consequently, the capability to display identified constraints visually would be of great

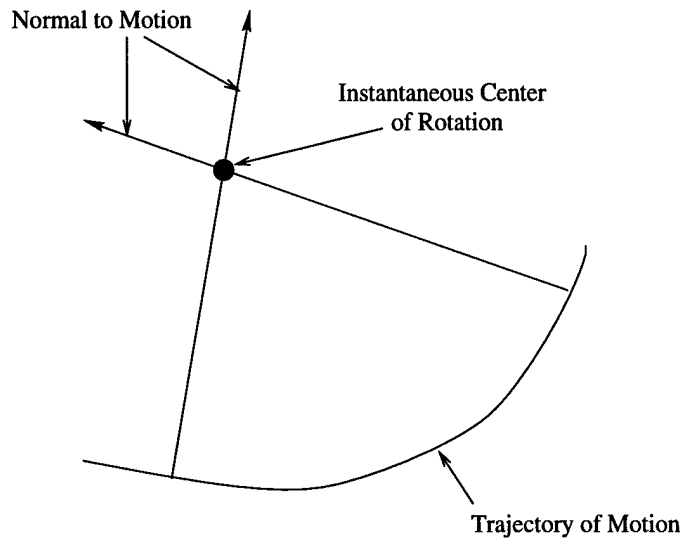


Figure 4.3: *Instantaneous Center of Rotation*

benefit. The best way to do this is to have the constraint representation in a format easily read by common computer graphics packages.

This brings up the issue of whether the representation should describe the system in geometric space or configuration space. The geometric space can be thought of as a traditional coordinate system where the manipulator arm and each object or constraint in the space is represented as geometric figures with physical shapes. In configuration space each point represents a complete set of coordinates describing a single configuration of position and orientation of the manipulator arm. The points in the space make up the possible configurations. The manipulator then is represented by one point corresponding to its configuration at any given time. Objects or constraints are described as the subset of points where the constraint acts on the manipulator. The advantage of the geometric space is that it gives a nice physical representation of objects, however, the configuration space description has the advantage of easily determining if contact is made or broken.

Another issue is that the representation must provide the graphical picture of the object,

its trajectory, and the corresponding constraint forces in a succinct and compact form. Due to memory limitation problems, only a limited amount of data can be stored. A good representation then, will fully describe the constraint and require as little information and memory space as possible.

Chapter 5

Experimental Results

Example tasks were performed and analyzed to demonstrate the parameter and constraint identification processes. The tasks were performed using a two-finger teleoperated planar hand [15] which is described below. A block stacking task was used to demonstrate the process of parameter identification, while a crank turning task and a needle puncture task demonstrated the constraint identification process. A virtual block and virtual crank were then created and calibrated using the information obtained from the example tasks. This chapter begins by describing the apparatus used in the example tasks, and then expounds on the details of these implementations. It concludes by describing the virtual world calibration that was performed.

5.1 Teleoperated Dextrous Hand System

These experiments use a planar, two-finger teleoperated hand with finger-tip force feedback [15]. This system trades a limitation on the number of joints for a clean and simple me-

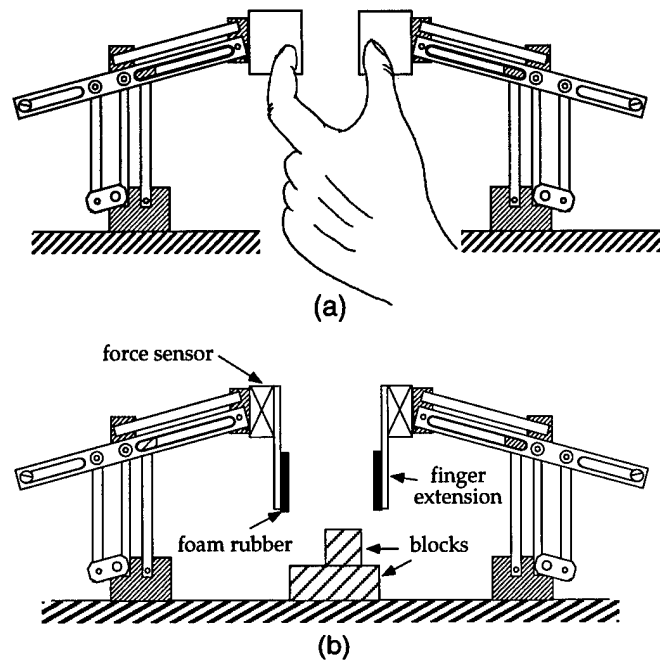


Figure 5.1: (a) *Master Manipulator with Operator's Hand.* (b) *Remote Manipulator.*

chanical design. The system has high bandwidth and a large dynamic range which permits accurate control of contact forces and small motions. The system is designed to execute tasks that humans usually accomplish with a precision pinch grasp between the thumb and index finger. For most tasks, the operator's wrist rests on the table top and the operator makes contact with the master only at the tips of the fingers (Fig. 5.1). The master and remote manipulators are kinematically identical, with two degrees of freedom in each finger, so finger tip position or force can be controlled within the vertical plane. The workspace is roughly circular and 75 mm in diameter. Parallelogram linkages maintain a constant vertical orientation of the finger tips, which preclude inappropriate torques on the operator's finger tips as the joints rotate. Two-axis strain gauge force sensors measure finger tip forces on both master and remote manipulator hands.

The controller uses a conventional bilateral force reflection control scheme. The joint angles of the master manipulator are the command inputs for position control of the joints of the remote manipulator, and the forces measured at the remote manipulator finger tips are the command inputs for force control of the master. Each finger is capable of applying a continuous tip force of at least 4 N. Flat, thin finger tips extending downward are mounted on the two remote manipulator fingers to facilitate manipulation of the rectangular blocks and other objects used in the experiments (Fig. 5.1b). The manipulator finger tips were covered with a 2 mm layer of closed-cell foam rubber to increase compliance and friction.

The trials for this case study were performed by the author after practicing sufficiently to become proficient at the task. Six sensor signals were recorded for each of the two remote manipulator fingers: two joint angles, two joint velocities, and horizontal and vertical components of finger tip force. The signals were collected at 50 ms intervals during the course of the task, for a total of 10 seconds. The forward kinematic relations permitted calculation of endpoint position and velocity of each finger, as shown in the subsequent figures.

5.2 Parameter Identification: Block Stacking Task

Pick-and-place tasks are a convenient starting point for the study of automatic identification techniques because the grasping and lifting actions that comprise these tasks are an essential part of many telemanipulation operations. These tasks are also amenable to automatic identification, as the parameters of interest and task segments are readily defined.

To describe the process of property identification, this section analyzes the block stacking

task introduced in section 3.1, which requires reversing the positions of two aluminum blocks stacked one atop the other. The top block is moved off of the stack and onto the ground, and then the other block is placed on top of it. Figure 3.1 shows the progression of the task through the various subtasks. It is assumed that the positions and velocities of the fingers performing this task and the force applied at the finger tips are all known.

5.2.1 Task Decomposition

Using the method described in section 3.1, the task was decomposed into subtasks and states. Recall the subtasks in order were *pregrasp of block 1*, *pickup*, *carry*, *place*, and *release* of the first block followed by *pickup*, *carry*, *place*, and *release* of the second block. The possible states were *free-motion*, *hold-on-ground*, *hold-above-ground*, and *hold-on-ground-with-sliding*. Table 5.1 shows the parameters that can be identified during each subtask and the associated state. It also shows the sensor signals required for estimating the environment parameters. Most of the parameters pertain to the first block (with the exception of the height of the lower block). Properties of the second block (and the height of the first block) can be estimated during the last five subtasks.

5.2.2 Segmentation

Segmentation proceeded using the modified Qualitative Reasoning method described in section 3.2. Recall that the stages of segmentation were transformation and thresholding, determination of states, and determination of sub-tasks. The transformation of data resulted in the following pertinent task motions and forces:

Table 5.1: *Relevant Signals and System States for Estimation of Object Parameters During the First Four Subtasks*

Subtasks	System State	Object Parameter			
		Weight Block #1	Width Block #1	Height Block #2	Friction of Fingers Against Block #1
Pregrasp, Block #1	free-motion				
Pickup, Block #1	hold-on-ground			vertical positions	horizontal and vertical forces
Carry, Block #1	hold-above- ground	vertical forces	horizontal positions		horizontal and vertical forces
Place Block #1	hold-on-ground			vertical positions	horizontal and vertical forces

Rigid Body Position = Mean of Finger Positions

Grip Width = Difference of Horizontal Finger Positions

Rigid Body Velocity = Average of Finger Velocities

Net Force on Block = Sum of Finger Forces

Grip Normal Force = Min. of Abs. Values of Horizontal Finger Force

Grip Shear (tangential) Force = Half of Difference of Vertical Finger Forces

Once the task motions and forces were obtained, the velocity and force data were passed through thresholding filters with equal positive and negative thresholds. The velocity thresholds were ± 5 cm/sec, and the force thresholds were ± 0.05 N. These threshold values were chosen because they fulfill the requirements that noise in the data stream does not cause the signal value to surpass the threshold, but significant changes in position and force do cause the value to surpass the threshold. Position, velocity, and force data for one trial of the stacking task are shown below in Figs. 5.2, 5.3, and 5.4. The progression of the

Table 5.2: *Qualitative Values of Sensed Parameters which Define Task States*

State	Sensed Parameters					
	Velocity Horizontal	Velocity Vertical	Average Horizontal Force	Average Vertical Force	Grip Force	Shear Force
<i>free-motion</i>			0	0	0	0
<i>hold-on-ground</i>		0		– or 0	+	
<i>hold-on-ground, sliding</i>	+ or –	0		– or 0	+	
<i>hold-above-ground</i>				+	+	

task can be discerned in these figures. The side to side pattern of motion can be seen in Fig. 5.2a, up and down motion in Fig. 5.2b, and grasp and release in Figure 5.2c. The vertical offset of the two fingers (Fig. 5.2d) is not of interest in this task, but has been included for completeness. In Fig. 5.3, the average velocities of the two fingers are shown, along with the results of thresholding. Note that velocities are close to zero during the pickup and place subtasks. The forces applied to the environment are shown in Fig. 5.4. Horizontal forces (Fig. 5.4a) are generally zero during the carry and free-motion subtasks, and non-zero during pickup and placement subtasks. Net vertical force (Fig. 5.4b) is also zero during free-motion, but positive during carry subtasks, and negative or zero during pickup and placement. The holding subtasks are clearly visible in Fig. 5.4c as large, positive grip forces.

These correlations between sensor data and task states were formalized into rules for automatically identifying the various states, as discussed in chapter 3 and summarized again here in Table 5.2.

Data for several trials of the block-stacking task were processed using the automatic state identification procedure. The segmentation algorithm divided the data into sections

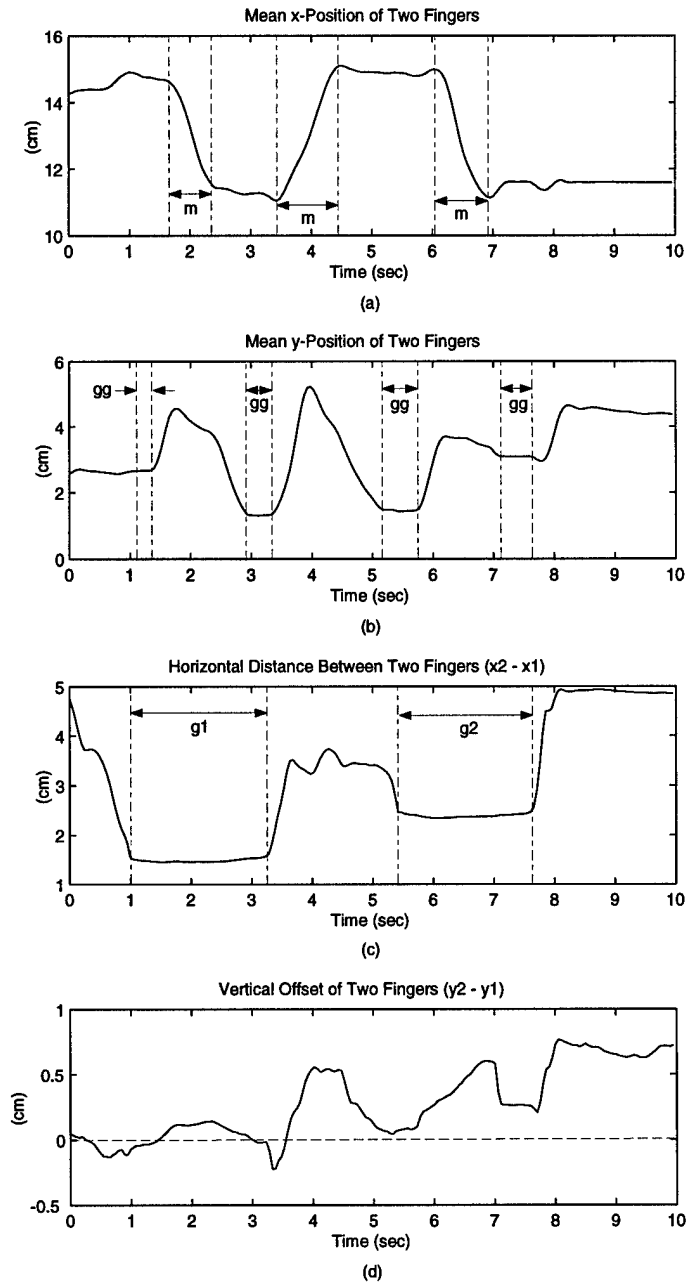


Figure 5.2: Horizontal and Vertical Positions of the Two Fingers. Legend: *m* = Movement of fingers to a new position; *gg* = Grasping an object on the ground; *g1* = Grasping object 1; *g2* = grasping object 2.

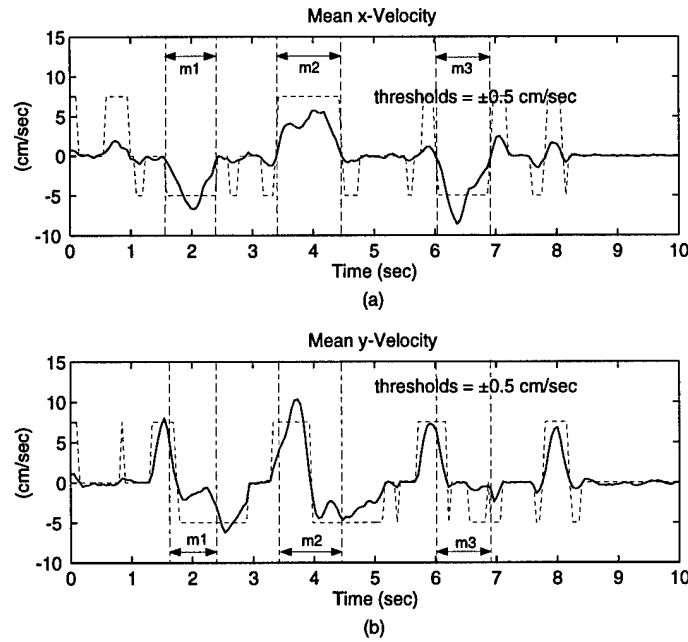


Figure 5.3: Mean (a) Horizontal and (b) Vertical Velocities of the Two Fingers. Legend: *m1* = Move block 1 to a new position; *m2* = Move fingers back to get block 2; *m3* = Move block 2 to a new position.

which corresponded closely to those selected by hand. For these data sets, the results of thresholding were not especially sensitive to the threshold values, although selecting appropriate thresholds can be difficult when noisy manipulators and a larger range of tasks and operators are involved [12].

Figure 5.5 shows the results of applying the procedure to the data in the preceding figures. The value of the plotted function denotes the identified state of the system at that time. A value of 1 corresponds to free-motion, a value of 2 to hold-above-ground, and a value of 3 to hold-on-ground. Times at which sliding-on-ground was identified are labeled with an "x". A value of zero indicates that none of the four possible states was identified. Accordingly, data from these times were not used in the identification of any object properties. These "dropouts" are caused mainly by transients in the signals just

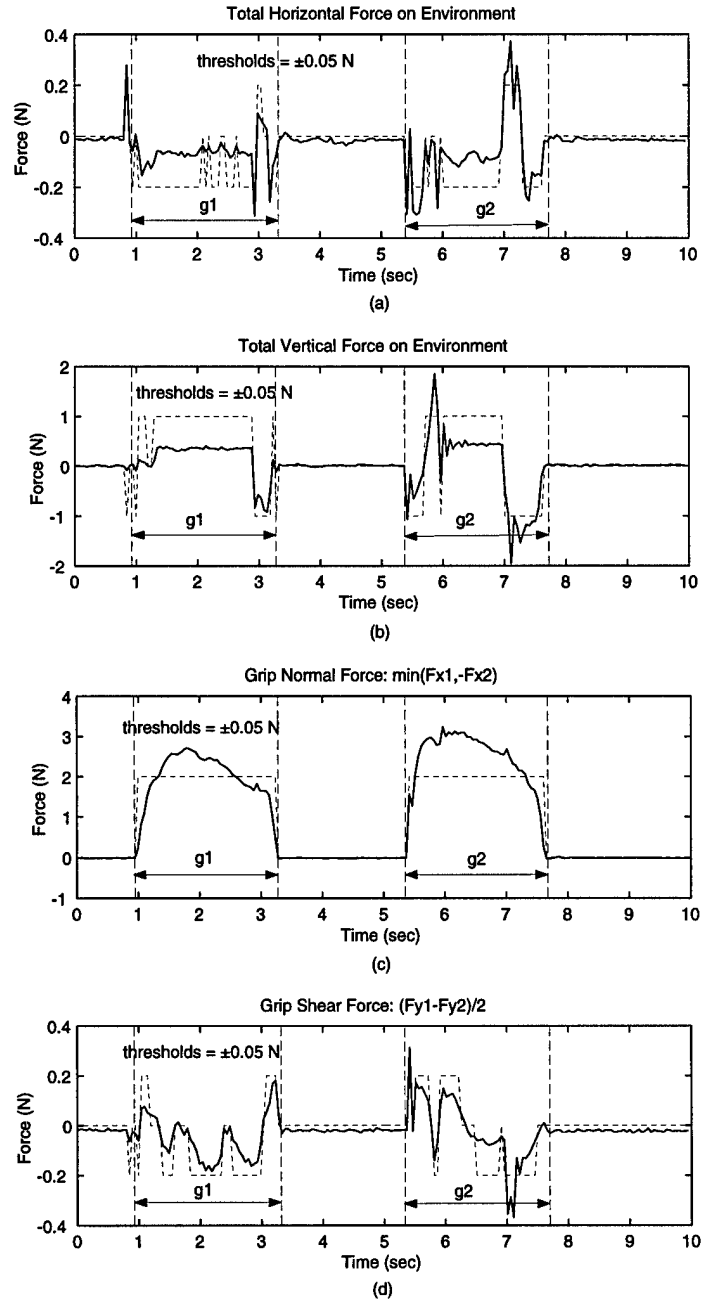


Figure 5.4: Forces: (a) Total Horizontal Force on Environment; (b) Total Vertical Force on Environment; (c) Grip Force; (d) Vertical Shear Force. Legend: g1 = Grasping object 1; g2 = Grasping object 2.

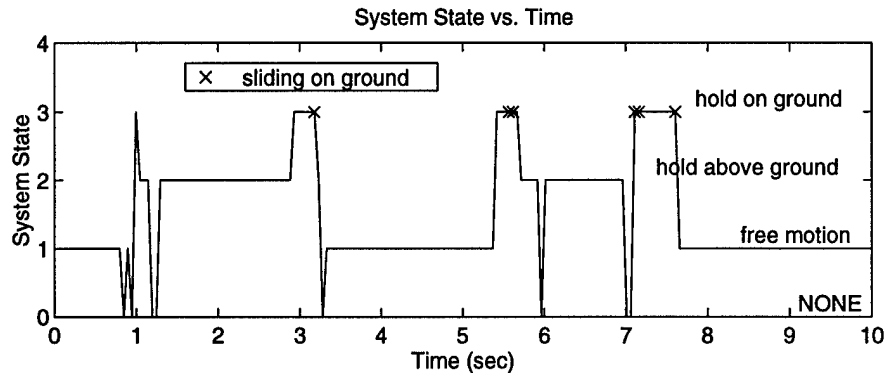


Figure 5.5: *Result of Automatic System State Identification Procedure.*

after lift-off or impact of the blocks. While these unidentified points are not desirable, they are not a problem as long as at least some data samples are successfully identified for each subtask. Using this convention, the expected (and observed) progression of states for this task was: 1, 3, 2, 3, 1, 3, 2, 3, 1. Subtasks I-IX were assigned to the data stream from the state information using the list of the anticipated order of the states.

5.2.3 Parameter Estimation

To illustrate the process of parameter estimation, four properties of the block from the example task were determined, although there are other properties that could have been estimated. The properties determined were block weight, height, width, and μ , the coefficient of static friction between the block and the fingers of the robot hand. Table 5.3 lists each property along with the subtasks in which the appropriate measurements can be collected. The last column of the table contains the simple formulas used to estimate the object properties. Because our goal is to outline the steps of the identification process and illustrate the key issues, no attempt has been made to find an optimal estimator. This table describes object property estimation during manipulation of the first block (Subtasks

II-IV); for estimation of the analogous properties during manipulation of the second block (Subtasks IV-VIII), each instance of “block 1” in the table is simply replaced by “block 2” and vice versa.

Ideally, the weight of block 1 was the sum of the vertical forces measured at each finger tip during the “carry block 1” subtask. The estimate was the average sum over all samples in the subtask (i.e. over N_{III} , the number of samples in subtask III). The weight estimate is sensitive to segmentation boundaries due to noise during the transition period between subtasks. Therefore, the weight estimate excluded force data during the first and last 5% of the carry subtask.

The width of each block is simply the average horizontal distance between the finger tips over the entire carry subtask. In contrast, the height estimate uses interactions between objects in the remote environment to determine the desired parameters. This requires multiple estimation steps because it is formed from two discrete measurements. For example, the height of block 2 is found from the difference between the vertical position of the fingers when block 1 is lifted from atop block 2 and when it (block 1) is subsequently placed on the table. Similarly, moving block 2 gives the height of block 1.

Note that for these experiments, we assume static friction can be modeled as a single constant μ . Ideally, μ would be measured at the onset of slip, but since slippage of the blocks between the fingers was not detectable, only a lower bound of μ was obtained. If slip did occur, the maximum value of μ recorded should correspond to the coefficient of static friction, recorded at the onset of slip. In many telemanipulation tasks, operators expressly avoid slips, so accurate determination of μ could require a special calibration procedure in which the operator deliberately causes the block to slip.

Table 5.3: Methods of Property Estimation

<i>Property</i>	<i>Subtask</i>	<i>System State</i>	<i>Formula for Estimate</i>
Weight (Block 1)	Subtask III Carry Block 1	hold-above-ground	$W = \Sigma \frac{F_{y1} + F_{y2}}{N_{III}}$ (Average sum of vertical forces for finger1 and finger2)
Height (Block 2) Part a:	Subtask II Pick up Block 1	hold-on-ground	$y_{II} = \Sigma \frac{y_2}{N_{II}}$ (Average vertical position of right finger)
Part b:	Subtask IV Place Block 1	hold-on-ground	$y_{IV} = \Sigma \frac{y_2}{N_{IV}}$ (Average vertical position of right finger)
Final:			$h = y_{II} - y_{IV} $ (Absolute value of the difference in average height)
Width (Block 1)	Subtask III Carry Block 1	hold-above-ground	$w = \Sigma \frac{x_2 - x_1}{N_{III}}$ (average difference in horizontal position)
μ (Block 1)	Subtasks II-IV Pick up Block 1, Carry Block 1, and Place Block 1	hold-on-ground and hold-above-ground	$\mu \geq \max \left \frac{F_{y1}}{F_{x1}} \right $ (Max. absolute value of vertical force over horizontal force)

Table 5.4: Comparison of Estimated (Mean and Standard Deviation) and Actual Environment Parameters

Property	Block 1		Block 2	
	Estimated Value	Actual Value	Estimated Value	Actual Value
Weight	0.352 ± 0.024 N	0.343 ± 0.0005 N	0.425 ± 0.071 N	0.437 ± 0.0005 N
Height	1.75 ± 0.01 cm	1.60 ± 0.01 cm	1.29 ± 0.02 cm	1.27 ± 0.01 cm
Width	1.47 ± 0.01 cm	1.59 ± 0.01 cm	2.36 ± 0.01 cm	2.54 ± 0.01 cm
Lower Bound on μ	0.49	—	1.06	—

5.2.4 Results of Parameter Identification for Block Stacking Task

The estimated parameters for one trial of the block-stacking task are listed in Table 5.4, along with actual values, as measured by a laboratory balance and calipers. Applying the algorithm to a different trial of the block stacking task gave similar estimates of the parameters. Certain trials exhibited significant vibrations following state transitions suggesting the need for data filtering prior to segmentation.

Figures 5.6, 5.7, 5.8, and 5.9 are plots of the measurements used to estimate object weight, height, width, and μ respectively. The portions of the data that were used in the estimates are indicated by solid lines. Figure 5.6 shows the force measurements used to find the average weight during the hold subtask. Note that there are two intervals of measurements, indicating that two objects were held. The large oscillations that occur during the second interval are due to inertial forces and the object impacting the surface. The average estimates of the weights are within 3% of the actual values. The standard deviation is relatively large for block 2 due to the oscillations in the force signals.

The first two solid lines in Figure 5.7 show the heights during the two hold-on-ground states, when block 1 is moved from the top of block 2 to the ground. The difference in these

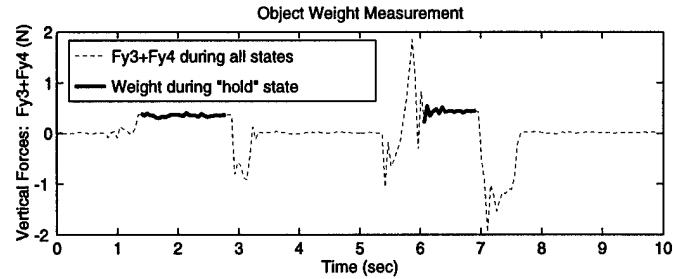


Figure 5.6: *Object Weights.* The solid portions are the segments of data used for weight estimation.

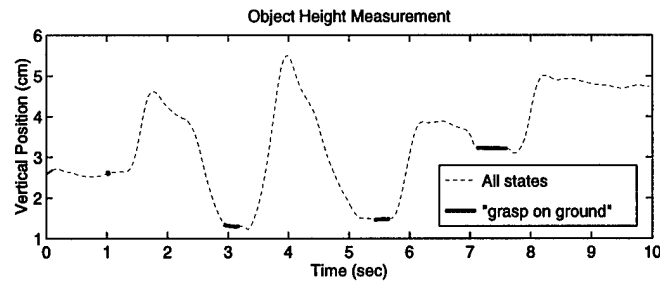


Figure 5.7: *Object Heights.* The solid portions are the segments of data used for height estimation. Object 2 height is the difference between the first and second solid lines, while object 1 height is the difference between the third and fourth solid lines.

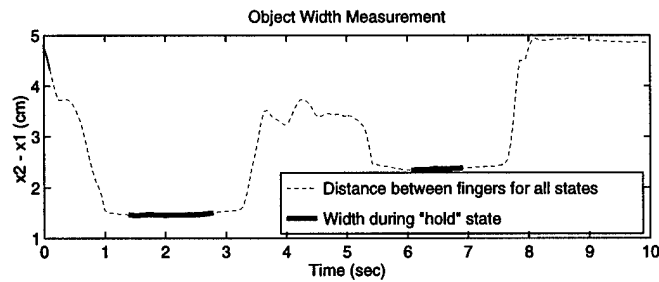


Figure 5.8: *Object Widths.* The two solid lines indicate the widths of the blocks.

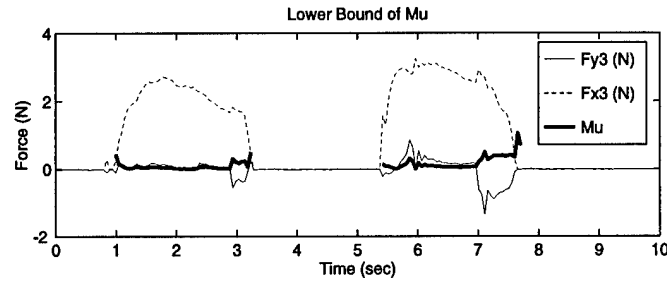


Figure 5.9: *Lower Bound of μ , Vertical Force F_{y3} , and Horizontal Force F_{x3} . μ is only calculated during “hold on ground” and “hold above ground” states.*

heights gives the height of the block originally beneath it. Likewise, the last set of two solid lines represents grasping and moving block 2 from the ground to the top of block 1. Notice that the intervals for these subtasks are fairly short, but even though there are only a few points in the interval, the estimated heights of blocks 1 and 2 are within 10% and 2% of their respective measured dimensions. The estimated height of block 1 shows greater error, perhaps due to slipping of the block within the fingers at the instant when the block makes contact with the ground. The solid lines in Fig. 5.8 show the widths of the two blocks. The errors in the estimates of the widths of the blocks were approximately 7%. Since fingertip position measurements are based on joint angle sensor signals, this discrepancy may be a result of compliance of the remote manipulator fingertips, or mis-calibration of manipulator kinematics. The solid line in Fig. 5.9 is the value of the ratio of Vertical Force to Horizontal Force, $\frac{F_y}{F_x}$, during the subtasks when the fingers are grasping the object. Since there was no way of knowing whether slips occurred, the maximum value of $\frac{F_y}{F_x}$ from the plot gives a lower bound on μ . The only way to get the actual value of μ would have been if slipping had occurred and the value of $\frac{F_y}{F_x}$ had been recorded at the moment sliding began.

5.3 Constraint Identification

To demonstrate the process of constraint identification, two example tasks involving constraints were performed. In the first, the teleoperated hand grasps a crank that is constrained to move in a circle and rotates it several times. The constraint identification process determined the existence of the constraint and then found its position and the associated constraint forces. The second example was a puncture task where the hand grasped a hypodermic needle and used it to puncture a simulated membrane. In this case, the membrane was the bottom of a paper cup. The constraint identification process determined the position and stiffness of this non-rigid constraint.

5.3.1 Crank Turning: Identification of a Rigid Constraint

In this first example, the planar two-fingered hand is used to turn a crank whose motion is constrained to lie on a circle (recorded trajectories are shown in Figure 5.10). Examination of the data indicates that the motions were essentially quasi-static and inertial terms were negligible. Furthermore, the task decomposition (i.e., our a priori knowledge of the expected forces) specifies that the component of the net force applied to the manipulated object which is normal to the direction of motion, F_n , will be dominated by gravity, $(mg)_n$. Any remaining component of F_n will be due to a constraint force, F_C .

$$F_n = (mg)_n + F_C \quad (5.1)$$

The gravity forces acting on the crank were identified from the forces tangential to the motion. From the task decomposition the force on the block tangential to the direction of

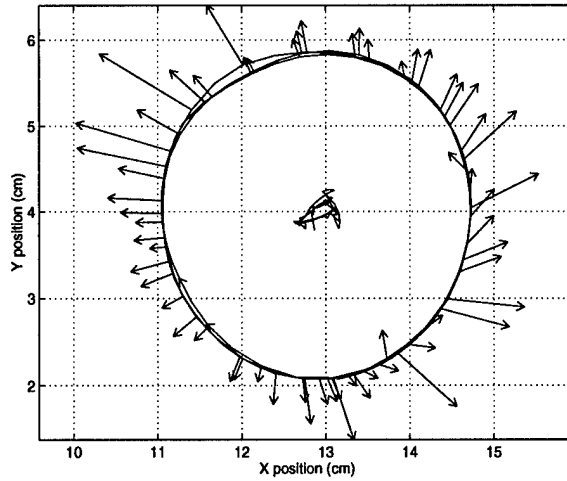


Figure 5.10: *Position of Crank Handle, Vectors of Applied Force, and Computed Center of Rotation. The vectors show direction and magnitude of force applied by the robot hand.*

motion, F_t , is expected to be of the form

$$F_t = (mg)_t + \mu F_n + cv \quad (5.2)$$

where $(mg)_t$ is the component of gravity tangential to the motion, μ is a coefficient of friction, c is a viscous coefficient and v is the object velocity. In Figure 5.11, F_t for the crank is plotted along with its least squares approximation which was used to find the values of the parameters of the constraint.

To test for the presence of constraints, $F_C = F_n - (mg)_n$ is plotted for the crank motions in Figure 5.12. As a comparison, $F_C = F_n - (mg)_n$ is also plotted for a portion of the block stacking task described in section 5.2. As expected, the constraint force on the block is negligible during its free motion while a significant positive constraint force can be seen throughout for the crank. The sign of the constraint force indicates that the operator was pulling radially outward on the crank while turning it. This fact is also evidenced by the direction of the force vectors in Fig. 5.10. While the crank represents a bilateral radial constraint, the motion of the operator was insufficient to fully capture both radial

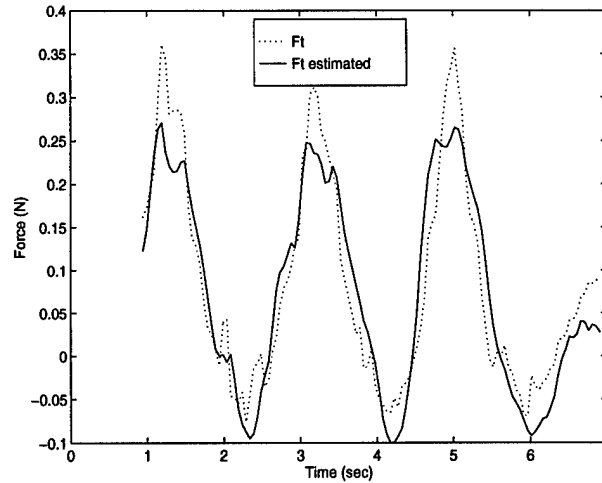


Figure 5.11: *Tangential Force, F_t , for the Crank and the Least Squares Approximation Given by $mg = 0.16$ N, $\mu = 0.053$ and $c = 0.010$.*

directions of the constraint. This example shows how exploratory motions or repeated trials by different operators may be necessary to fully identify a constraint.

The crank handle and bearings are much stiffer than the manipulator finger pads. In addition, the constraint is active for all configurations of the crank. Thus, Rule (1) is employed from section 4.3.2 and a single screw of zero pitch is used to describe the constraint. The location of the screw axis in the plane of motion is equivalent to the center of rotation. The center of rotation is computed by intersecting normals to the instantaneous direction of motion for 12 consecutive data points and then finding their mean position. Figure 5.10 depicts the position of the crank handle computed as the average displacement of the fingers as well as the center of rotation for three rotations of the crank and the force vectors applied by the fingers. Drift in the location of the screw axis is largely due to slippage between the fingers and the crank handle. Finger stiffness is relatively unimportant here, as it only accounts for displacements of about 0.11 cm. Note that if the fingers were grasping the coupler link of a four-bar mechanism, the location in the plane of the screw axis would be

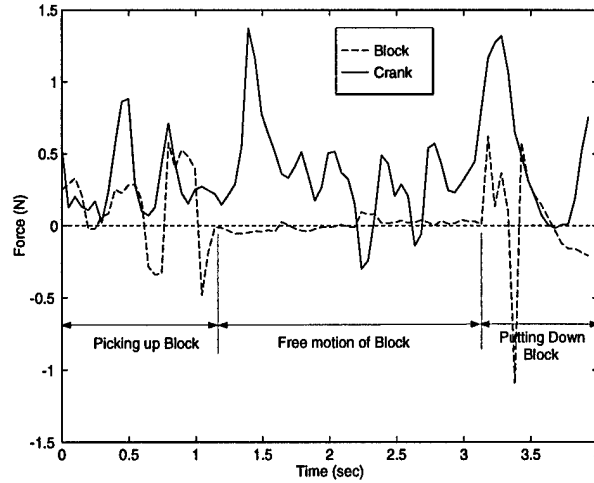


Figure 5.12: *Constraint Forces, F_C , for the Block and Crank.*

configuration dependent. Also note that the applied force when moving upward (right half of circle) has a distinct tangential component, but when moving downward the applied force has a very small tangential component. This is due to the fact that gravity and friction forces are additive while moving up but almost negate each other while going down.

5.3.2 Needle Insertion: Identification of a Configuration Dependent Non-rigid Constraint

In this example, a hypodermic needle was inserted through the bottom of an inverted paper cup as shown in Figure 4.2 using the manipulator of Figure 5.1. Since the constraints are both configuration and history dependent, they are modeled using Rule 2 of section 4.3.2. In particular, we focus on the constraint force in the direction of needle insertion which is associated with screw s_2 . The screw axis itself can be located using position and velocity data from the fingers.

As in the previous example, the insertion motion was quasi-static. Since the constraint

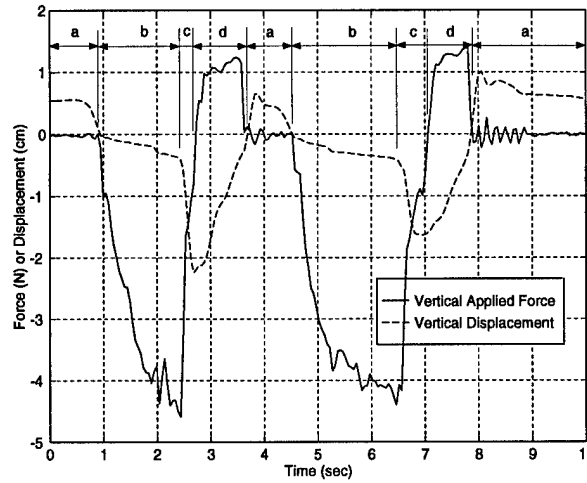


Figure 5.13: *Constraint Force and Displacement in Insertion Direction. a = free motion, b = insertion, c = penetration, d = extraction.*

force lies in the direction of motion, the tangential force equation becomes

$$F_t = (mg)_t + F_C \quad (5.3)$$

the force F_C is plotted along with tangential displacement in Figure 5.13. The subtasks of free motion, needle insertion, penetration and extraction can be clearly seen in the data.

The history dependence of the constraint force arises from two factors. First, the force depends on the maximum prior insertion depth. This can be seen by the fact that the extraction force is much smaller than the insertion force. Secondly, the cup bottom acts as an elastic membrane in series with the elastic finger pads. During extraction, the constraint force depends on the displacement of both from their relaxed positions. A model of constraint stiffness can be obtained by dividing the constraint force by the position at each point in time. The history dependent stiffness as a function of time is plotted in Figure 5.14.

Repeated trials with direction reversals at various depths would allow for the identification of a more detailed model.

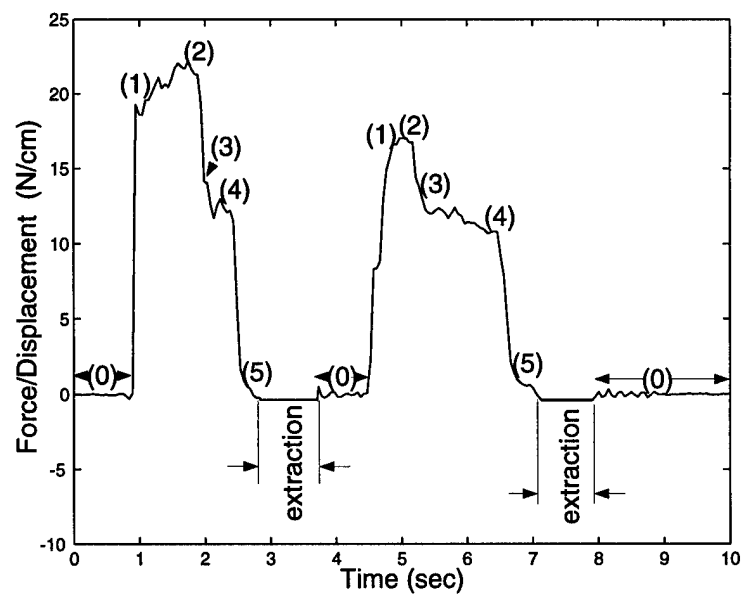
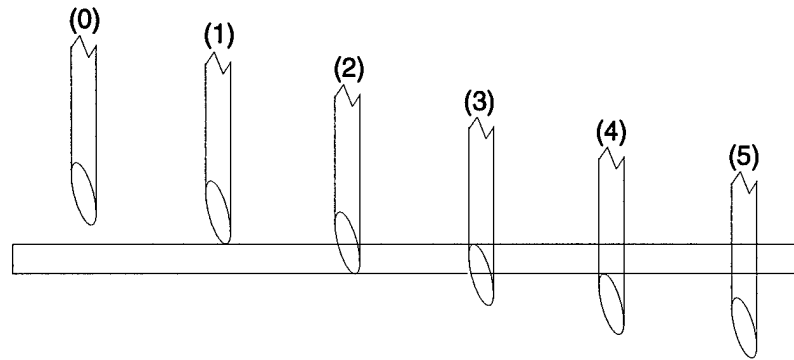


Figure 5.14: *Model of History Dependent Stiffness. The position of the needle tip corresponding to each number on the plot shows configuration information at key transition points in the insertion process.*

5.4 Development and Calibration of Virtual Model

A virtual block and virtual crank were calibrated using parameters and constraints identified in the previous section. The virtual objects were simulated using a program that displays an arbitrary virtual block on a computer screen and provides haptic feedback to the operator through the master hand. The code was modified so that it could accept the identified parameters and constraint information resulting in a calibrated virtual block and virtual crank. A print out of the the pertinent subroutines for determining the forces to be applied to the fingers and the motions of the virtual blocks are given in the appendix. This section describes how the teleoperated hand system and control code were modified to display virtual objects and it describes the details of the virtual model calibration.

5.4.1 Modification of Two-fingered Hand System

The same planar, two-finger teleoperated hand system described in section 5.1 was used but instead of the master hand controlling the remote manipulator, it controlled a virtual hand with virtual fingers on a computer screen. The remote manipulator was taken completely out of the loop. As the virtual fingers contacted objects on the screen, reaction forces on the fingers were calculated and imposed on the real fingers while contact forces acting on the virtual objects and their resulting motions were calculated and animated on the screen. Figure 5.15 shows the interactions of this system. Based on a description of the task the operator sends commands via the master controller to the simulation being run on a computer chip, which then computes reaction forces and motions. These motions are displayed on the monitor while reaction forces act on the operator's fingers via the master

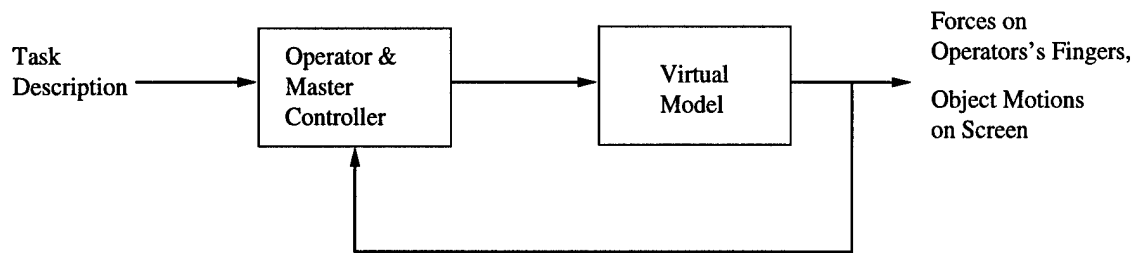


Figure 5.15: *Block Diagram Showing Interactions Between the Operator and the Virtual Environment*

manipulator.

The contact and reaction forces were calculated using velocities and positions determined by an Euler integration method. While there are other more accurate numerical integration methods, Euler integration was employed due to its speed. Integration error accumulation over multiple time steps was minimized since accelerations, velocities, positions, and reaction forces were calculated based on measured finger forces at each time step rather than from values computed at previous time steps.

5.4.2 Block Calibration

A virtual block was calibrated using the weight, height, and width parameters identified for one of the real blocks (block 2) in section 5.2.4. Figure 5.16 depicts the teleoperated hand during block calibration. A simulation of the the calibrated block was then implemented using the master manipulator and a video monitor (See Fig. 5.17). As the operator moves the fingers on the master, the simulation displays the fingers on the computer screen. As the virtual fingers contact the virtual block, reaction forces of the block acting on the fingers are computed and displayed through the master manipulator on the operator's hand. The operator then is able to both see the interactions of the virtual block with the fingers on

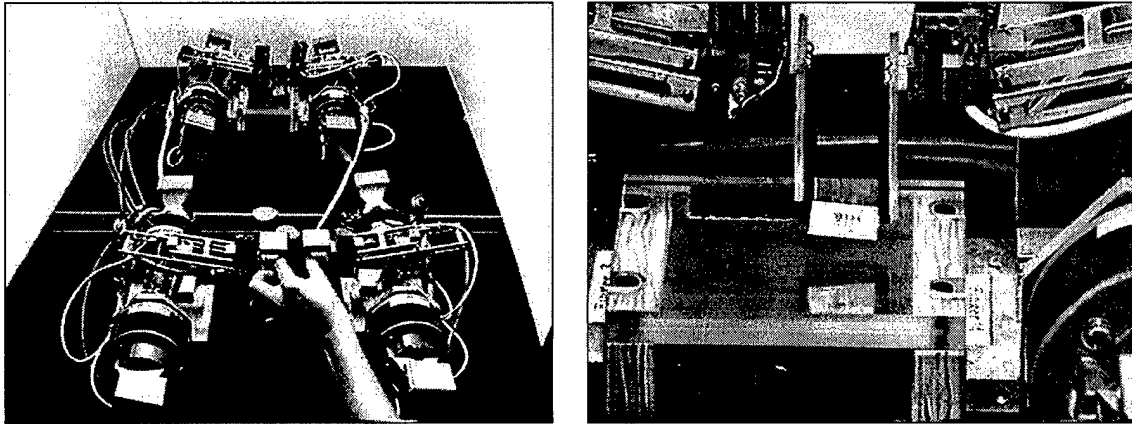


Figure 5.16: *Master and Remote Manipulators (left) and Close-up of Remote Manipulator (right) During Block Calibration.*

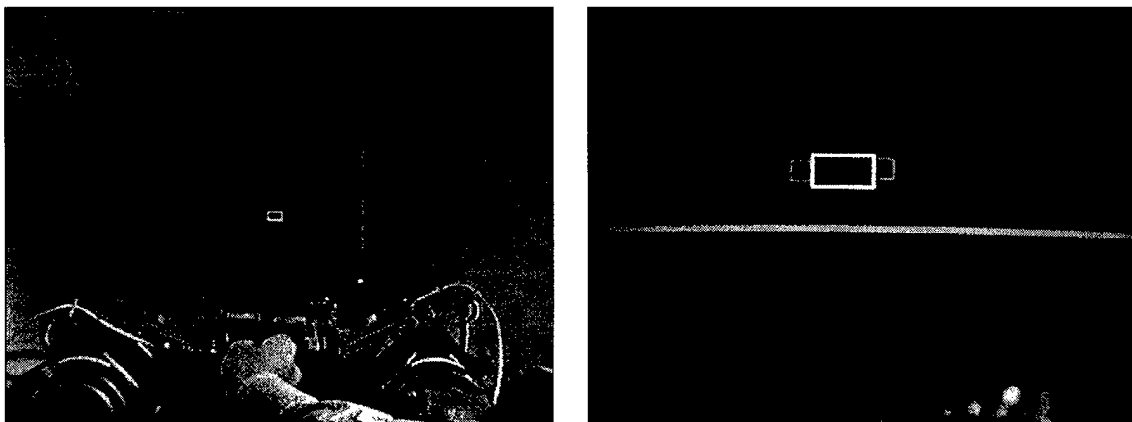


Figure 5.17: *Virtual Simulation of Calibrated Block.*

the screen and feel the forces of the block acting on his hand.

The virtual block on the screen appeared with the same size and shape as the one used in the real block stacking task. As the operator manipulated the master controller causing the virtual fingers to contact the virtual block, the reaction forces made the operator feel as though he were manipulating a block very similar in size, shape, and mass to the real block. The operator was able to lift and maneuver the block around the screen in the same way the real block had been manipulated. The virtual forces from the operator interactions

with the virtual block, (See Figs. 5.18 and 5.19), can be compared with those from the real block (See Fig. 5.4). The similarity in the force plots shows that the operator experienced comparable forces during interactions with the virtual block and the real block. The Grip Normal Force appears is smaller for the virtual block appears smaller because the operator didn't happen to squeeze as hard as he did for the real block. The Virtual Grip Shear Force does not show the same positive to negative jumps because the operator tried to avoid differences in in finger vertical forces that created system instabilities. This instability was a result of a stick-slip condition that occurs as one finger pushes up and the other pushes down. It is important to note that the virtual block was picked up twice during the time interval during which data was collected and it corresponds to the second block that was manipulated during interactions with the real blocks.

Figure 5.20 shows a plot of the sum of the vertical forces during manipulation of the virtual block. The time intervals during which the block is held above the ground are shown as solid lines and correspond to a weight measurement of the block. Comparing this to Fig. 5.6, one can see the virtual block produced approximately the same weight forces acting on the operator's fingers as the real block (Block 2) did. Figure 5.21 shows a plot of the distance between the two fingers during the virtual simulation. The solid portion indicates the time interval during which the block is being grasped and corresponds to the width of the block. A comparison of this plot to the one in Fig. 5.8 shows that the width of the simulated block is very close to that of the real block (Block 2).

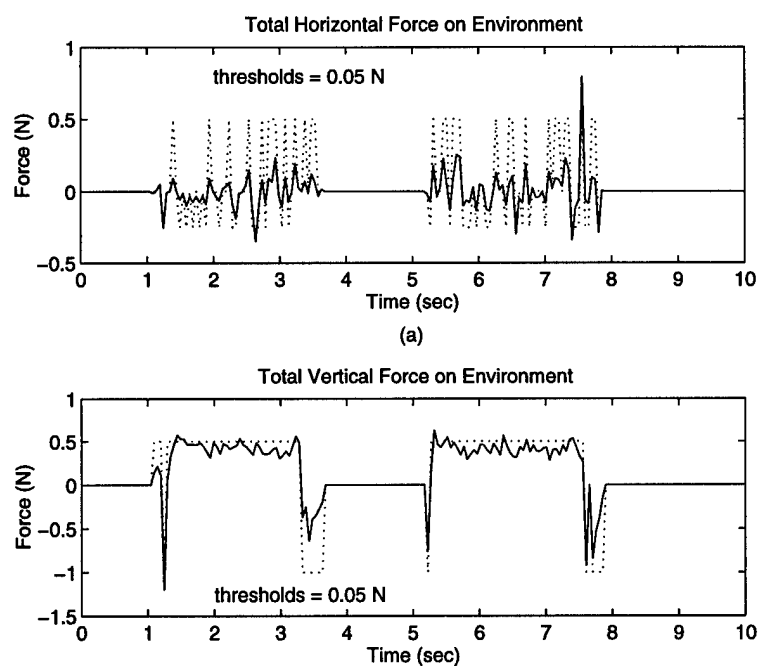


Figure 5.18: (a) *Horizontal Forces Acting on Virtual Block.* (b) *Vertical Forces Acting on Virtual Block.*

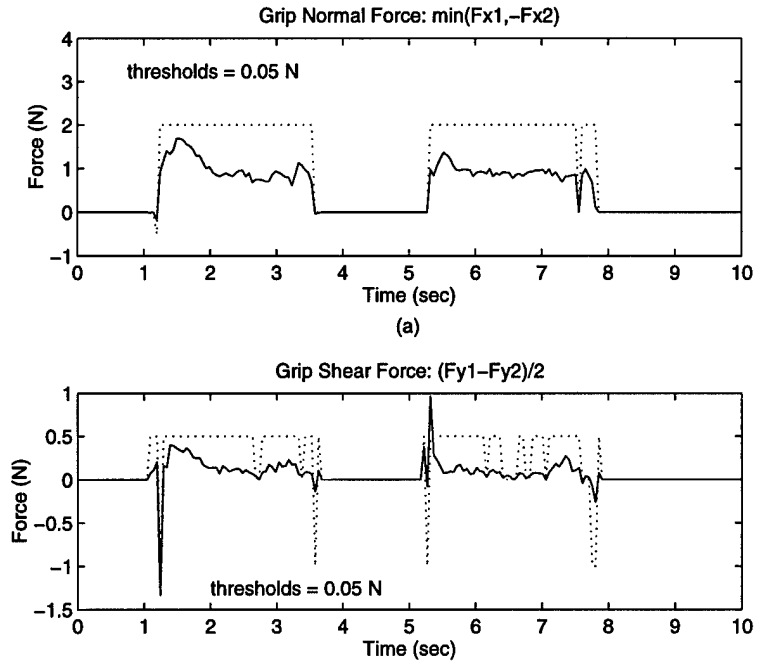


Figure 5.19: (a) Grip Force Acting on Virtual Block. (b) Grip Shear (vertical) Force Acting on Virtual Block.

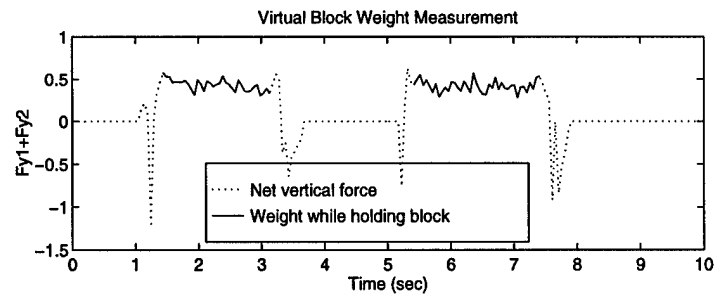


Figure 5.20: Virtual Block Weight Measurement. The solid portions are the intervals during which the virtual block is being held above ground.

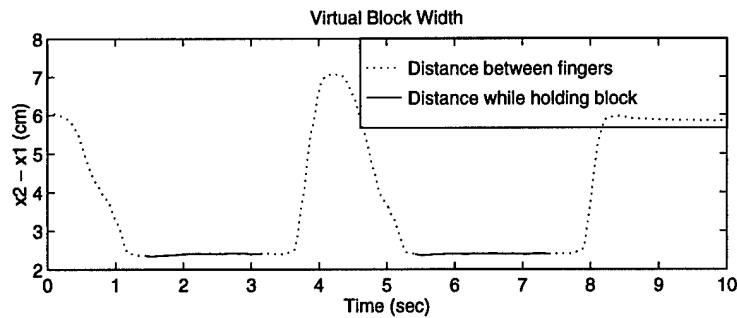


Figure 5.21: *Virtual Block Width Measurement. The solid portions correspond to the intervals during which the virtual block is held above ground.*

5.4.3 Crank Calibration

The goal in calibrating a virtual crank was to obtain one that exhibited the same rigid bilateral constraints as the real crank. Using the model and parameters identified in section 5.3.1, the constraint forces of the virtual crank were programmed so that the fingers grasping the virtual crank handle followed the appropriate circular trajectory. The position of the real crank screw axis, the radius of the trajectory, and the rigidity of the constraint previously identified were used in calibrating the virtual crank. As the operator manipulated the virtual crank with the master, he could see the handle move in a circular trajectory and feel the forces constraining the handle to move in a circular motion. Figure 5.22 shows the the trajectory, center of rotation, and constraint forces of the virtual crank. The virtual crank was constrained to move in the same location and with the same radius as the real crank shown in Fig. 5.10. The forces for the virtual crank appear to be greater than those for the real crank as the it moves around the top half of its trajectory. This results from the fact that the operator happened to be pushing with greater force while manipulating the the virtual crank. The center of rotation for the virtual crank appears to wander around the

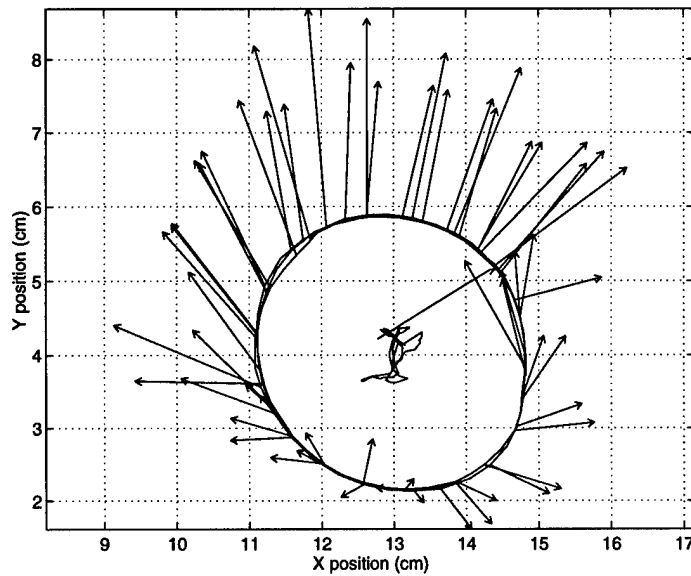


Figure 5.22: *Position of Virtual Crank Handle, Vectors of Applied Force, and Computed Center of rotation. The vectors show the direction and magnitude of force applied by the virtual fingers on the virtual crank.*

center point in a manner similar to that of the real crank. However, there is one portion where the center of rotation appears to move to the upper right. This corresponds to a point in time where the operator moved very slowly in the tangential direction so that small displacements in the constraint direction made a larger contribution to the center of rotation calculations. The stiffness of the crank is close to the real crank but not identical because stiffness is limited by stability issues.

Chapter 6

Conclusions

6.1 Summary of Thesis

This thesis addresses the problem of how to automatically identify remote object properties and constraints in a teleoperated robot system and how to use those parameters to calibrate virtual environment models. To do this, a parameter identification process was developed and applied to teleoperated tasks to determine the parameters in question. The process consists of three parts namely task decomposition, segmentation, and parameter estimation. In the case of constraint identification, this third part includes identifying the existence of the constraint, modeling the constraint and estimating constraint parameters. These identified parameters were then used in the creation of a virtual environment that replicates the real environment.

This thesis has presented the following new accomplishments:

- Development and implementation of a method to identify object parameters and constraint parameters to be used in calibration of remote environment models. This was

performed automatically using data collected from a teleoperated task. It required the use of a three part parameter estimation algorithm in which the task was decomposed into subtasks, the data segmented into pertinent time intervals, and the desired parameters identified. While portions of this algorithm have been developed by others, they were used separately and for other purposes. This is the first time an algorithm integrating the three parts has been developed.

- Distinction between subtasks and states during task decomposition process. This distinction and the determination of when subtasks occur provides additional insight into task progression.
- Automatic identification of constraint existence, constraint parameters, and modeling of constraints without assuming prior knowledge of the constraint. No assumption was made about the form of the constraint or about the parameters that fit the model. This makes the process more general and allows it to be applied to a greater class of problems.
- Calibration of virtual models using data from experiments performed on real objects. This resulted in virtual models with the same dimensions and force properties as the real objects without the time consuming iterative refinement process required when using arbitrary parameters for virtual models.

6.2 Future Work

The work presented can be enhanced with future work in the following areas:

- *Integration of the three part estimation process.* This involves developing the method presented here into an autonomous system for parameter identification. This system could be further developed into an on-line estimator.
- *Development of specific applications.* Part of this process is determination of the set of tasks and subtasks required for a given application and the specification of a complete set of parameters to be identified. In the Toxic Waste Remediation application, the set of tasks and subtasks would be composed of variations of the block stacking task described in this thesis. A surgical training application would require the development of new sets of tasks including cutting tissue and palpation of tissue.
- *Determination of time-varying properties.* There are many issues involved in how to handle time variance. One is determination of the proper data fitting technique to use when trying to estimate a changing value. Another issue is that the model would have to be updated frequently in order to accurately represent the changes. A solution might be to build the time rate of change of the parameter directly into the model.
- *Design of a user friendly interface for interactive calibration.* To find the best possible estimate of properties, there is a need to have interaction between the estimator and the user to determine when enough data has been collected and what data still needs to be collected. This could be done either by having the estimator determine when there is enough data to meet model fidelity requirements or by having the operator try out a virtual simulation of the model to see if it "feels" right. If either the estimator or the operator declares the model needs further refinement, additional calibration would then be performed.

- Design of the “assistant” that uses identified parameters to provide task information to the user. An issue involved here is determination of the most useful information and the best way to display it to the user. A balance must be maintained between providing too little useful information versus overloading the operator with information beyond his capability to comprehend it all.

Appendix A

Control Code for Virtual Simulation

The control code used to run the virtual simulation of the calibrated crank and block has two parts. The first part runs on a DSP chip inside a PC, and computes the forces and motions due to the interactions between the hand and the virtual crank, and displays these forces at the master manipulator interface. The main file for this force display code is `VROBOT.C`. This main file calls functions contained in the included file `BLOCK2.C`.

The second part of the code runs on the PC itself. It is responsible for loading `VROBOT` into the DSP chip, and then for running the visual display of the simulation on the computer screen. The main file for this image display code is `RUNBLOK2.C`.

The portions of the code of most interest to the work presented in this thesis, are those which determine the forces and motions of the virtual objects. Because of space limitations this appendix contains only the portions of these files containing these pertinent pieces of

code. Section A.1 contains the main body of the program `VROBOT.C` Section A.2 contains only the interesting subroutines of `BLOCK2.C` that are called by `VROBOT.C`.

A.1 VROBOT.C

This file computes the forces to be displayed at the master controller during the virtual simulation. The main body of this file is given here.

```

/*****
/* Program Name: vrobot.c
/* Parris Wellman
/* Modified from robot.c
/* Further Modifications by Tim Schulteis
/*
/* This program runs on the DSP chip and determines appropriate
/* torques to be applied to motors to run the virtual simulation
/* of calibrated objects.
/*
/* An interface that provides gravity compensation for the master
/* and feeds back arbitrary forces that were determined on the PC
/* Uses DMA in dual ported memory to obtain force values and to
/* Write back joint angles for the PC to use.
/* Note the absolute addressing scheme (borrowed from LSI)
/*
/* Last Modified : 1/18/97
/*
*****/
/*****

main() {
block = (OBJECT *) malloc(sizeof(OBJECT));
cyclecounter = 0;

/* Initialize C30, set A/D sample rate */
*PRIMCTL_REG = PRIMWD; /* Initialize the primary bus control register */
*EXPCCTL_REG = EXPWD; /* Initialize the secondary bus control register*/

/* Initialize AD BOARD */
*AD_CONTROL = AD_CHO; /* Setup control register */
*AD_TIMER = TIMER_VALUE; /* Setup timer on AD board */

reset_motors();
/* Assign control parameters*/
for (i=0;i<4;i++)
{fingerforce[i] = 0.0;}
weight.b1=MGL_11; /*Weight of bottom joint finger 1*/
weight.t1=MGL_12; /*Weight of top joint finger 1 */
weight.b2=MGL_21; /*Weight of bottom joint finger 2*/
weight.t2=MGL_22; /*Weight of top joint finger 2 */

spring.f1=TRACKFORCE_1; /*Bias force (g) to follow finger 1*/
spring.f2=TRACKFORCE_2; /*Bias force (g) to follow finger 2*/

k.p1=k_p1; /* bottom joint position gain */
k.p2=k_p2; /* top joint position gain */
k.v1=k_v1; /* bottom joint velocity gain */
k.v2=k_v2; /* top joint position gain */
k.fx=k_fx; /* x direction force gain */
k.fy=k_fy; /* y direction force gain */
k.fvx=k_fvx; /* x direction force damping gain*/
k.fvy=k_fvy; /* y direction force damping gain*/

```

```

k.v1 = k.v2 = 0.0;
dat = get_data();
read_force_offsets(0);
vel_off = read_vel_offsets();
init_block(block);
while (1) {
dat = get_data();      /*take raw data from sensors*/
ang = read_ang(0);     /*calculate position angles in degrees*/
/*read velocities of master only*/
vel = read_vel(0);

calc_sin(0);           /*Calculate sines of master only      */
cosine=calc_cos(0);    /*Calculate cosines of master only */
calc_cosdbl(0);        /*Calculate the torque correction
for the top axis*/

/* Read forces of master only      */
force      = read_forces(0);

forces[0] = force.x1/GRAMSTONEWTNS;
forces[1] = force.y1/GRAMSTONEWTNS;
forces[2] = force.x2/GRAMSTONEWTNS;
forces[3] = force.y2/GRAMSTONEWTNS;

get_positions();

/*calculate the forces to be played on the master fingers*/
playsound = fingerforces(position,block,forces,velocity,fingerforce,test);

/*calculate the vibration amplitude at impact, and update model */
leftamplitude = onestimestep(block,floorK,floorKV,test);
rightamplitude = leftamplitude;

/*holdfingers(); this is a check function */

/*send the positions of the block and fingers to the virtual */
/*world model running on the 486 */
send_positions();

/*these are the forces that we want the user to exert*/
/*fingerforces[#] are read from the virtual model.  */
forcef.x1 = fingerforce[0];
forcef.y1 = fingerforce[1];
forcef.x2 = fingerforce[2];
forcef.y2 = fingerforce[3];

/*****

/* NOTE: Forces given to the control law are
the ones we want the USER to exert.      */
/* Control law for finger 1*/

tor.t1 = (1/torcor.t1)*(weight.t1*cosine.t1
- forcef.y1*(L/100.0)*cosine.t1
+ forcef.x1*(L/100.0)*sine.t1);
tor.b1= weight.b1*cosine.b1
- forcef.y1*(L/100.0)*cosine.b1
+ forcef.x1*(L/100.0)*sine.b1;

/* Control for finger 2 */

tor.t2 = (1/torcor.t2)*(weight.t2*cosine.t2
- forcef.y2*(L/100.0)*cosine.t2

```

```

+ forcef.x2*(L/100.0)*sine.t2);
tor.b2 = weight.b2*cosine.b2
- forcef.y2*(L/100.0)*cosine.b2
+ forcef.x2*(L/100.0)*sine.b2;

/*No Slave in this Implementation*/
tor.t3 = 0;
tor.b3 = 0;
tor.t4 = 0;
tor.b4 = 0;

move_motors(0, tor);
    /* if (fabs(block->v.y) > T00_FAST)
{flop = 1.0;}
else flop = 0.0;    */
flop = (flop) ? 0 : 1;          /* used to see servorate from */
*DSP_A = (0x4000*flop << 16); /* DSP channel A (pin 7)      */
*DSP_TRIG = 0;

out(15,flop*10);
out(14,forces[0]);

cyclecounter++;
}
}

```

A.2 BLOCK2.C

This file contains functions called by VROBOT.C. Only the functions pertaining to force and motion determination are presented here.

```

/*****
/*unit for virtual crank experiments */
/*Parris Wellman Harvard University*/
/*Revised by Tim Schulte Boston University*/
/*Last Revision: 1/18/97 */
/* */
/*****
#include <math.h>
#include "block2.h"

/*****
/*Inputs: OBJECT *block int *leftamplitude int *rightamplitude */
/*Outputs: 1 if we are at initial contact and a wave should be played */
/* 0 otherwise */
/*Function: Does one Euler integration step and updates the values in */
/* block. */
/* Expects lengths in meters. */
/* Masses in kilograms. */
/* ONE_TIME_STEP = 1/servo-frequency */
/*****
float onestimestep(OBJECT *block,double floorK, double floorKV, double *test)
{
float dt;
vector2d acc;
vector2dn accn;
static int returnval;
float amplitude;
float impulse;
float denom;
float theta, thetadot; /*Angle of t,n coord system w.r.t. x,y*/
float co, si; /*Trig func. of theta*/
float phi; /*Angle of block w.r.t. center*/
float arg1, arg2; /*Arguments used in computing thetadot*/
float xdif, ydif; /*Abbreviation for x - xcenter*/

returnval = 0;
dt = ONE_TIME_STEP;

returnval = Floor(block,floorK,floorKV);

if (returnval == 1){
amplitude = block->v.y;
}
else{
amplitude = 0.0;
}

if (block->f.x > FORCEMAX) {block->f.x = FORCEMAX;}
if (block->f.x < (-FORCEMAX)) {block->f.x = -FORCEMAX;}
if (block->f.y > FORCEMAX) {block->f.y = 2*FORCEMAX;}
if (block->f.y < (-FORCEMAX)) {block->f.y = -2*FORCEMAX;}

/*Compute Theta (Angle of tan-dir relative to horizontal)*/
/* and Theta Dot */
xdif= block->r.x-XCENTER;

```

```

ydif= block->r.y-YCENTER;
arg1= (ydif)/(xdif);
arg2= (block->v.y*(xdif) - block->v.x*(ydif))
      / (xdif) / (xdif);
phi=atan (arg1);
if (xdif < 0)
    { phi += PI;
    }
theta = phi - PI/2;
thetadot= arg2/(1+arg1);
co= cos(theta);
si= sin(theta);

/* Assign trig func. to bus */
block->theta.co = co;
block->theta.si = si;

/*Compute the accelerations (GRAVITY accounted for in force calc) */
acc.x = block->f.x/block->M;
acc.y = block->f.y/block->M;

/*Compute the velocities */
block->v.x = block->v.x + acc.x*dt;
block->v.y = block->v.y + acc.y*dt;
block->vn.n = block->v.y*co
- ydif*si*thetadot
- block->v.x*si
- xdif*co*thetadot;

/*Compute the positions */
block->r.x = block->r.x + block->v.x*dt;
block->r.y = block->r.y + block->v.y*dt;
block->rn.n = -R + ydif*co - xdif*si;
block->rn.t = xdif*co + ydif*si;

return(amplitude);
}

/*****
/*Inputs:  double *angles OBJECT *block double *forces */
/*         double *vels double *fingerforce */
/*Outputs: none */
/*Function: computes the interaction forces based on a viscoelastic */
/*           model of the block. */
/*           Updates the forces applied on the block in *block. */
/*           Updates the forces applied to the fingers in *forces */
/*           *angles provides the joint angles of the fingers */
*****/
int fingerforces(double *pos,OBJECT *block,
double *forces,double *vel,
double *fingerforce, double *test)
{
    static float lastyleft;
    static float lastyright;
    static float lastblocky;

    int i,playleft;

    float x1,y1,x2,y2;
    float vx1,vy1,vx2,vy2;
    float kvy;

    float xleft,xforceleft;
    float xright,xforceright;
    float yleft,yright;

```

```

float vleft,vright;
float xvelforceright,xvelforceleft;
float fyleft,fyright;
float yforceleft,yforceright;
float mu1,mu2;
float co,si;          /*Cosine and Sine variables*/
float yfingfob,xfingfob; /*Net fingerforces acting on block*/
float constraintforce; /*Value of constraint force in constraint dir.*/

/*calculate the positions of the fingertips*/
x1 = (XBASE1 + pos[0] + HANDLEWIDTH);
y1 = (YBASE1 + pos[1]);
x2 = (XBASE2 + pos[2] - HANDLEWIDTH);
y2 = (YBASE2 + pos[3]);

/*calculate the xpositions of where block contact points should be*/
xleft = (block->r.x - block->w);
xright = (block->r.x + block->w);

/*ycontact position on block always at center*/
/* yleft = (block->r.y + (block->h/2));
   yright = (block->r.y + (block->h/2));
*/
yleft = (block->r.y);
yright = (block->r.y);

/*ycontact position on block stays where contact is first made */
/*contactleft(right) is distance from ref point (bottom of block) to */
/*point where contact was first made*/
/* if (xleft >= x1)
   {   block->contactleft = y1 - block->r.y;
   }
   if (x2 >= xright)
   {   block->contactright = y2 - block->r.y;
   }
   yleft = (block->r.y + block->contactleft);
   yright = (block->r.y + block->contactright);
*/

/*calculate the velocities*/
vx1 = vel[0];
vy1 = vel[1];
vx2 = vel[2];
vy2 = vel[3];

/*calculate the forces in the x direction on both fingers*/
xforceleft = -block->K*(x1 - xleft) - block->KV*(vx1-block->v.x);
xforceright = - block->K*(x2 - xright) - block->KV*(vx2-block->v.x);

/*calculate the forces in the y direction on both fingers*/
yforceleft = - block->KVERT*(y1 - yleft) - block->KVVERT*(vy1-block->v.y);
yforceright = - block->KVERT*(y2 - yright) - block->KVVERT*(vy2-block->v.y);

/* Forces are present only when fingers are in contact with block */
/* if (xleft > x1)
   {   xforceleft = 0;
   }
   yforceleft = 0;
   if (xright < x2)
   {   xforceright = 0;
   }
   yforceright = 0;
*/
*/

```

```

/*assign the forces to fingerforce*/
fingerforce[0] = (double) (xforceleft);
fingerforce[1] = (double) (yforceleft);
fingerforce[2] = (double) (xforceright);
fingerforce[3] = (double) (yforceright);

/*Eq. and opp. fingerforces, gravity, and constraint force act on block*/
xfingfob=-(xforceleft + xforceright); /*Net xfingerforce acting on block*/
yfingfob=-(yforceleft + yforceright); /*Net yfingerforce acting on block*/
/*if abs(block->rn.n-R) > .05 */
/*if (block->rn.n > R) */
if (1)
    { constraintforce = - (block->rn.n)*KCONSTRAINT
    - block->vn.n*KVCONSTRAINT ;
    }
else constraintforce = 0;
block->f.x = xfingfob - constraintforce*block->theta.si;
block->f.y = yfingfob + constraintforce*block->theta.co
- block->M*GRAVITY;
return(1);
}

```


Bibliography

- [1] Allen, P. K., 1990, "Mapping Haptic Exploratory Procedures to Multiple Shape Representations," *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, Vol. 3, May, pp. 1679-1684.
- [2] An, C., Atkeson, C., and Hollerbach, J., 1995, "Estimation of Inertial Parameters of Rigid Body Links of Manipulators," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, Vol. 2, December, pp. 990-995.
- [3] Atkeson, C., An, C., and Hollerbach, J., 1985, "Rigid Body Load Identification for Manipulators," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, Vol. 2, December, pp. 996-1002.
- [4] Bejczy, A. K., Kim, W. S., and Venema, S. C., 1990, "The Phantom Robot: Predictive Displays for Teleoperation with Time Delay," *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, Vol. 1, May, pp. 546-551.
- [5] Cutkosky, M. R. and Hyde, J. M., 1993, "Manipulation Control with Dynamic Tactile Sensing," *6th International Symposium on Robotics Research*, Hidden Valley, PA, October.
- [6] Delson, N. and West, H., 1996, "Segmentation of Task Into Subtasks for Robot Programming by Human Demonstration," *ASME Japan/USA Symposium on Flexible Automation*, Boston, MA, Vol. 1, July, pp. 41-47.
- [7] Driels, M. R., 1993, "Using Passive End-Point Motion Constraints to Calibrate Robot Manipulators," *Transactions of the ASME*, Vol. 115, September, pp. 560-566.
- [8] Fiorini, P., Losito, S., Giancaspro, A., and Pasquariello, G., 1992, "Neural Networks for Off-line Segmentation of Teleoperation Tasks," *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, Glasgow, UK, Vol. 1, August, pp. 17-22.
- [9] Funda, J., and Paul, R. P., 1991, "Efficient Control of a Robotic System for Time-Delayed Environments," *91 ICAR. Fifth International Conference on Advanced Robotics. Robots in Unstructured Environments.*, Pisa, Italy, Vol. 1, June, pp. 219-224.

- [10] Fyler, D., 1981, "Computer Graphic Representation of Remote Environments Using Position Tactile Sensors," SM thesis, MIT, August.
- [11] Griebenow, B.E., 1994, "Buried-waste Integrated Demonstration Retrieval Projects." Annual Meeting of American Nuclear Society, New Orleans, LA, 19-23 June, *Transactions of the American Nuclear Society*, Vol. 70, p. 402.
- [12] Hannaford, B., and Lee, P., 1991, "Hidden Markov Modal Analysis of Force/Torque Information in Telemanipulation," *International Journal of Robotics Research*, Vol. 10, No. 5, October, pp. 528-539.
- [13] Hannaford, B., Wood, L., Guggisberg, B., McAfee, D., and Zak, H., 1989, "Performance Evaluation of a Six-Axis Generalized Force-Reflecting Teleoperator," JPL Publication 89-18, Pasadena, CA.
- [14] Hannaford, B., Wood, L., McAfee, D. A., and Zak, H., 1991, "Performance Evaluation of a Six-Axis Generalized Force-Reflecting Teleoperator," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, May/June, pp. 620-633.
- [15] Howe, R. D., 1992, "A Force-Reflecting Teleoperated Hand System for the Study of Tactile Sensing in Precision Manipulation," *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, Vol. 2., May, pp. 1321-1326.
- [16] Kang, S. B., and Ikeuchi, K., 1993, "Toward Automatic Robot Instruction from Perception — Recognizing a Grasp From Observation," *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 4, pp. 432-443.
- [17] Kondoleon, A. S., 1976, *Application of Technology-Economic Model of Assembly Techniques to Programmable Assembly Machine Configuration*, S.M. Thesis, MIT Mechanical Engineering Department.
- [18] Khosla, P., and Kanade, T., 1985, "Parameter Identification of Robot Dynamics," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, December, pp. 1754-1760.
- [19] Leahy, M. B., Jr. and Hamill, N., 1995, "The Next Generation Munitions Handler Prototype Acquisition Campaign: Targets & Courses of Action," Research Paper, ACSC/DEA/205/05-05, Air Command and Staff College.
- [20] Lin, S. T., and Yae, K. H., 1992, "Identification of Unknown Payload and Environmental Parameters for Robot Compliant Motion," *Proceedings of the 1992 American Control Conference*, Chicago, IL, Vol. 4, June, pp. 2952-2956.
- [21] Ljung, L., 1987, *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ.
- [22] MacLean, K., 1986, "The 'Haptic Camera': A Technique for Characterizing and Playing Back Haptic Properties of Real Environments," *Proceedings of the ASME Dynamic Systems and Control Division*, Atlanta, GA, DSC-Vol. 58, November, pp. 459-467.

- [23] McCarragher, B. J., 1994, "Force Sensing from Human Demonstration Using a Hybrid Dynamical Model and Qualitative Reasoning," *Proceedings of the 1994 IEEE Conference on Robotics and Automation*, San Diego, CA, Vol. 1, May, pp. 557-563.
- [24] McCarragher, B. J., 1994, "Petri Net Modeling for Robotic Assembly and Trajectory Planning," *IEEE Transactions on Industrial Electronics*, Vol. 41, No. 6, December, pp. 631-640.
- [25] McCarragher, B.J., and Asada, H., 1993, "Qualitative Template Matching Using Dynamic Process Models for State Transition Recognition of Robotic Assembly," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 115, June, pp. 261-269.
- [26] Nevins, J. L. and Whitney, D. E., 1980, "Assembly Research," *Automatica*, Vol. 16, pp. 595-613.
- [27] Noyes, M. V. and Sheridan, T. B. 1984. "A Novel Predictor for Telemanipulation through a Time Delay," *Proceedings of the Annual Conference on Manual Control*, Moffett Field, CA, NASA Ames Research Center.
- [28] Okamura, A. M., Turner, M. L., and Cutkosky, M. R., 1997, "Haptic Exploration of Object with Rolling and Sliding," *Proceedings of the 1997 International Conference on Robotics and Automation*, Albuquerque, NM, submitted.
- [29] Pook, P. K., and Ballard, D. H., 1993, "Recognizing Teleoperated Manipulations," *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, Vol. 2, May, pp. 578-585.
- [30] Rabiner, L.R., 1989, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, February, pp. 257-285.
- [31] Rabiner, L.R. and Juang, B. H., 1986, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, Vol. 3, No. 1, January, pp. 4-16.
- [32] Satava, R. M., 1993, "Virtual Reality Surgical Simulator," *Surgical Endoscopy*, Vol. 7, No. 3, May-June, pp. 203-205.
- [33] Satava, R. M. and Simon, I. B., "Teleoperation, Telerobotics, and Telepresence in Surgery," *Endoscopic Surgery and Allied Technologies*, Vol. 1, No. 3, June, pp. 151-153.
- [34] Schulteis, T. M., Dupont, P. E., Millman P. A., and Howe, R. D., 1996, "Automatic Identification of Remote Environments," *Proceedings of the ASME Dynamic Systems and Control Division*, Atlanta, GA, DSC-Vol. 58, November, pp. 451-458.
- [35] Strang, G., 1988, *Linear Algebra and it's Applications*, Harcourt Brace Jovanovich, San Diego, CA.
- [36] Yang, J., Xu, Y., and Chen, C. S., 1994, "Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics," *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 5, October, pp. 621-631.